

PKI/CA与数字证书 技术大全

张明德 刘 伟 编著



中国工信出版集团



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
<http://www.phei.com.cn>

内 容 简 介

数字证书技术，又称作PKI技术或CA技术，不仅涉及的技术领域广泛、标准规范庞杂，而且还涉及运营管理和法律法规；本书是中国第一部全面介绍数字证书技术的书籍，涵盖技术、标准、运营、法规等内容。为方便读者快速理解PKI、快速把握数字证书技术，并能快速运用到具体的工作当中，本书主要从七个方面来全面介绍数字证书技术，主要内容包括：如何理解PKI、PKI技术基础、PKI之数字证书与私钥（网络身份证）、PKI之CA与KMC（管理网络身份证）、PKI之应用（使用网络身份证）、PKI之运营（CA中心）、PKI之法规与标准。

本书精心选材、内容翔实、重点突出、特点鲜明，既有原理介绍，又有实验案例，具有很强的实用性。可以作为从事信息安全领域（如系统设计、软件开发、项目实施、系统运维、技术管理等）的技术人员的技术参考手册，也可以作为希望了解数字证书技术的各类企事业单位技术人员或管理人员的学习资料，同时还可以作为信息安全、密码学、计算机等专业的本科高年级学生和研究生入门教材。

策划编辑：赵 平

责任编辑：周宏敏

封面设计：一克米工作室



ISBN 978-7-121-26106-0



9 787121 261060 >

定价：98.00元

PKI/CA与数字证书

技术大全

第1版 (2010年11月) 第1次印刷

中国铁道出版社



中国铁道出版社



中国铁道出版社

PKI/CA 与数字证书 技术大全

张明德 刘 伟 编著

电子工业出版社

Publishing House of Electronics Industry

北京 • BEIJING

内 容 简 介

数字证书技术, 又称作 PKI 技术或 CA 技术, 不仅涉及的技术领域广泛、标准规范庞杂, 而且还涉及运营管理和法律法规。本书是国内第一部全面介绍数字证书技术的书籍, 涵盖技术、标准、运营、法规等内容。为方便读者快速理解 PKI、快速把握数字证书技术, 并能快速运用到具体的工作当中, 本书主要从七个方面来全面介绍数字证书技术, 主要内容包括: 如何理解 PKI、PKI 技术基础、PKI 之数字证书与私钥(网络身份证)、PKI 之 CA 与 KMC (管理网络身份证)、PKI 之应用(使用网络身份证)、PKI 之运营(CA 中心)、PKI 之法规与标准。

本书精心选材、内容翔实、重点突出、特点鲜明, 既有原理介绍, 又有实验案例, 具有很强的实用性。可以作为从事信息安全领域(如系统设计、软件研发、项目实施、系统运维、技术管理等)的技术人员的技术参考手册, 也可以作为希望了解数字证书技术的各类企事业单位技术人员或管理人员的学习资料, 同时还可以作为信息安全、密码学、计算机等专业的本科高年级学生和研究生入门教材。

未经许可, 不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有, 侵权必究。

图书在版编目(CIP)数据

PKI/CA 与数字证书技术大全/张明德, 刘伟编著. —北京: 电子工业出版社, 2015. 6
ISBN 978-7-121-26106-0

I. ①P… II. ①张… ②刘… III. ①计算机网络—安全技术 IV. ①TP393.08

中国版本图书馆 CIP 数据核字(2015)第 105944 号

策划编辑: 赵 平

责任编辑: 周宏敏

印 刷: 北京京科印刷有限公司

装 订: 北京京科印刷有限公司

出版发行: 电子工业出版社

北京市海淀区万寿路 173 信箱 邮编 100036

开 本: 787×1 092 1/16 印张: 33 字数: 888 千字

版 次: 2015 年 6 月第 1 版

印 次: 2015 年 6 月第 1 次印刷

印 数: 3000 册 定价: 98.00 元

凡所购买电子工业出版社图书有缺损问题, 请向购买书店调换。若书店售缺, 请与本社发行部联系, 联系及邮购电话: (010) 88254888。

质量投诉请发邮件至 zltsp@phei.com.cn 盗版侵权举报请发邮件至 dbqq@phei.com.cn

服务热线: (010) 88258888。

前 言

(一)

四十年前，世界上还没有公钥密码算法。1976 年，公钥密码算法的思想被提出，1978 年，第一个公钥密码算法 RSA 诞生了，标志着密码学进入了一个全新时代，使密码技术的应用从单纯的保密通信扩展到了身份认证。

三十年前，世界上还没有数字证书。1988 年，第一个数字证书标准 X.509 v1 诞生了（作为 ITU X.500 的一部分），标志着密码学应用开始进入 PKI 时代。

二十年前，中国还没有数字证书。1997 年，第一个基于数字证书的电子交易规范 SET 诞生了，当年便正式走进中国。由于受电子商务泡沫经济和对 PKI 前景过于追捧的影响，自 1998 年开始形成一种盲目的 CA 中心建设热潮，各部委及各省份（直辖市）均开始积极投资建设 CA 中心。

十年以前，大家都不知道数字证书（俗称 U 盾）为何物，就连专门从事数字证书服务的 CA 中心也不知道路在何方。截至 2004 年底，一半以上省份（直辖市）建立了区域 CA 中心，部分部委建立了行业 CA 中心，CA 中心总数已经超过 60 家，形成了一种乱序竞争的态势。尽管累计投入已超数亿元，但数字证书发放量不过几百万，业务收入不足几千万。随着电子商务泡沫的破灭，全球经济进入低谷，CA 中心也陷入困境。

五年以前，很多人依然不知道数字证书为何物，但已经有发烧友开始炫耀如何在家里就可实现网上银行汇款转账，也已经有企业员工开始享受在办公室就可完成报税、报关、报检等业务。2005 年 4 月 1 日开始生效的《电子签名法》，让数字证书有了法律的武装，也给 CA 中心带来了崭新的生机，CA 中心正式进入规范管理和有序发展的轨道，数字证书也逐步在报税、报关、报检、网上银行等领域得到广泛应用。截至 2010 年底，获得行政许可资质的 CA 中心约 30 家。

到了今天，还有几个人不知道数字证书为何物呢？没有数字证书，你还对淘宝网店的资金账户放心吗？你还敢开通网上银行的汇款转账功能吗？你还能忍受去报税或社保大厅去排队办理业务吗？已经有调皮者开始抱怨手里的 U 盾太多！哎，只能怪他银行账户太多。截至 2013 年底，获得行政许可资质的 CA 中心约 33 家，有效数字证书已达 2.6 亿张。CA 中心的发展进入黄金时期，数字证书进一步向社保缴纳、公积金管理、第三方支付、电子病历、移动办公、企业管理、电子保单、网上招投标等领域渗透。

也许五年以后，“一证通”时代就会来到。一个 U 盾就可访问所有银行的网上银行，一张数字证书就能访问政府的所有公共业务，我们拭目以待！

(二)

那么数字证书到底为何物？PKI 和 CA 又是什么？数字证书与公钥密码算法有什么关

系？既然有了数字证书，为什么还需要私钥？既然数字证书是公开的，还需要U盾干什么？很多人对此都困惑不解。由于数字证书涉及的技术领域非常广泛，而且技术专业性强，如果没有相当的专业功底，指望在短期内快速搞清楚相关概念和基本原理是十分困难的。

通俗地讲，数字证书可想象成一个“网络版的身份证”。数字证书里包含姓名、性别等身份信息，更重要的是也包含一个公钥。每个用户都拥有一个数字证书和一个私钥（与数字证书中的公钥配对），数字证书可以公开，但私钥必须保密。基于数字证书和私钥，素昧平生的交易双方无需见面，在网上就可确认对方的真实身份（客户需出示自己的数字证书，商家首先从客户数字证书中获得其身份信息，然后商家远程验证客户是否拥有对应的私钥；如果验证通过，则说明客户的身份真实有效，可以继续交易，否则应拒绝交易）。由于私钥的安全性至关重要，为防止私钥泄露，必须采用硬件设备加以安全保护，对于个人私钥，目前均采用USB Key方式，俗称为U盾。

专门负责颁发数字证书的系统称为CA系统，负责管理并运营CA系统的机构称作CA中心。所有与数字证书相关的各种概念和技术，统称为PKI（Public Key Infrastructure），其中数字证书格式、CA以及如何使用数字证书等内容均属于PKI范畴。PKI的主要功能是绑定证书持有者的身份和相关的密钥对（通过为公钥及相关的用户身份信息签发数字证书），为用户提供方便的证书申请、证书作废、证书获取、证书状态查询的途径，并利用数字证书及相关的各种服务（证书发布、黑名单发布、时间戳服务等）实现通信中各实体的身份认证、完整性、抗抵赖性和保密性。

但在日常沟通交流中，也将PKI技术称为CA技术、数字证书技术。

（三）

数字证书技术并不仅仅只解决身份认证问题，事实上，它已经成为当前解决身份认证、数据保密与完整、行为抗抵赖等问题的最佳技术。尽管数字证书如此重要、如此普及，但这方面的书籍并不多，要么过于简单内容不全面，要么偏重理论缺乏实用性，对行业内从事技术管理、系统设计、软件开发、项目实施、系统运维等人员的指导性不足。

本书作者具有近二十年的应用安全工作经验和近十年的企业信息化工作经验，见证了中国CA行业从无到有的发展历程，有丰富的PKI领域研究、开发、工程及标准等相关经验。为方便业界人士快速理解PKI、快速把握数字证书技术，并能快速运用到具体的工作当中，作者将多年的实践经验进行总结和提炼，经过长达近两年的辛苦编写终于完成本书的全部内容，希望能得到业内同行们的指教，以促进CA行业的健康发展。本书是国内第一部全面介绍PKI技术的书籍，涵盖技术、标准、运营、法规等内容。

本书内容共分为7部分，由29章内容组成：

第一部分：“如何理解PKI”。由3章内容组成，包括为什么会出现PKI技术、PKI包括哪些内容、其他非对称密钥管理体系等。

第二部分：“PKI技术基础”。由4章内容组成，包括ASN.1及其编码规则、密码技术、LDAP技术、实验一（DER编码示例和RSA算法示例）等。

第三部分：“PKI之数字证书与私钥：网络身份证”。由6章内容组成，包括公/私钥格式、数字证书格式、数字证书分类、私钥与证书存储方式、私钥与证书访问方式、实验二（RSA公钥格式编码示例、数字证书格式编码示例和Windows证书库操作示例）等。

第四部分：“PKI 之 CA 与 KMC：管理网络身份证”。由 5 章内容组成，包括系统结构、系统设计、对外在线服务、网络部署结构、实验三（OpenSSL CA 和 EJBCA 示例）等。

第五部分：“PKI 之应用：使用网络身份证”。由 4 章内容组成，包括基本应用、通用应用技术、常见应用、实验四（Windows IIS、Apache、Tomcat 等服务器证书配置）等。

第六部分：“PKI 之运营：CA 中心”。由 4 章内容组成，包括机房建设、运营文件、业务管理、资质申请等。

第七部分：“PKI 之法规与标准”。由 3 章内容组成，包括国内法规、国内标准、国际标准等。

本书精心选材、内容翔实、重点突出、特点鲜明，既有原理介绍，又有实验案例，具有很强的实用性。本书非常适合以下读者：

1. 从事信息安全领域（如系统设计、软件研发、项目实施、系统运维、技术管理等）的技术人员。

2. 希望了解数字证书技术的各类企事业技术人员或管理人员。

3. 信息安全、密码学、计算机等专业的本科高年级学生和研究生。

本书由张明德担任主编，刘伟等多人参与了本书部分内容的编写。其中，张明德负责内容策划、提纲拟定、统筹协调、内容审核和大部分内容的编写；刘伟负责第 6 章、第 12 章、第 18 章、第 19 章、第 22 章、第 24 章、第 29 章等内容的编写；张迪、刘文涛、胡安勇、李伟斌、王秋凤等参与部分内容的资料整理。本书在写作过程中得到了很多朋友的支持和关心，在此非常感谢所有关心、支持和帮助过作者的朋友们。

由于 PKI 是一门专业性很强的技术，涉及知识面很广，况且作者的能力及水平有限，出书时间又十分紧张，本书的缺点或错误在所难免，如蒙指正不胜感激。热忱欢迎广大读者的批评指导。

作者联系方式：zmdbook@163.com。

张明德

2014 年 9 月于北京

目 录

第一部分 如何理解 PKI

第 1 章 为什么会出现 PKI 技术	2
1.1 保密通信催生了密码技术	2
1.1.1 古代中国军队的保密通信方法	2
1.1.2 传统密码学与古代西方保密通信方法	3
1.1.3 两次世界大战的密码斗法	6
1.1.4 现代密码学与信息时代	7
1.2 密码技术普及推动了密钥管理技术的发展	9
1.2.1 密钥管理	9
1.2.2 对称密钥管理技术	11
1.2.3 非对称密码技术简化了密钥管理	12
1.3 PKI 本质是把非对称密钥管理标准化	14
1.4 私钥专有性使人联想到手写签名	15
1.5 电子签名法赋予电子签名与认证法律地位	16
第 2 章 PKI 包括哪些内容	18
2.1 PKI 体系框架	18
2.2 PKI/数字证书与私钥	20
2.3 PKI/CA 与 KMC	22
2.4 PKI/应用	25
2.5 PKI/运营	27
2.6 PKI/法规与标准	29
2.6.1 国内法规	29
2.6.2 国内标准	30
2.6.3 国际标准	31
2.7 PKI/信任模型	36
2.7.1 根 CA 信任模型	37
2.7.2 交叉认证信任模型	38
2.7.3 桥 CA 信任模型	39
2.7.4 信任列表信任模型	39

第 3 章 其他非对称密钥管理体系	41
3.1 PGP	41
3.2 EMV	42

第二部分 PKI 技术基础

第 4 章 ASN.1 及其编码规则	48
4.1 ASN.1 (抽象文法描述语言)	48
4.2 BER (基本编码规则) 与 DER (定长编码规则)	50
4.2.1 数据类型标识	50
4.2.2 BER 基本编码规则	52
4.2.3 DER 定长编码规则	54
第 5 章 密码技术	56
5.1 密码算法	56
5.1.1 算法分类	56
5.1.2 对称密码算法	56
5.1.3 非对称密码算法	60
5.1.4 摘要算法	62
5.2 运算模式 (工作模式)	64
5.2.1 ECB	64
5.2.2 CBC	64
5.2.3 CFB	64
5.2.4 OFB	65
5.3 扩展机制	65
5.3.1 MAC 与 HMAC	65
5.3.2 OTP	67
5.3.3 数字签名	68
5.3.4 数字信封	69
5.4 密码应用实践	70
5.4.1 软件加密与硬件加密	70
5.4.2 网络层加密与应用层加密	72
5.4.3 密钥管理的基本原则	72
5.4.4 密码设备的自身安全性	73
5.5 密码算法 ASN.1 描述	74
5.5.1 密码算法格式	74
5.5.2 密码算法 OID	75
5.6 密码消息 ASN.1 描述	75
5.6.1 通用内容消息 ContentInfo	75
5.6.2 明文数据消息 Data	75

5.6.3	数字签名消息 SignedData	76
5.6.4	数字信封消息 EnvelopedData	77
5.6.5	数字签名及信封消息 SignedAndEnvelopedData	78
5.6.6	摘要消息 DigestedData	78
5.6.7	加密数据消息 EncryptedData	79
5.6.8	密钥协商消息 KeyAgreementInfo	79
5.6.9	密码消息类型 OID	79
5.7	Base64 编码	80
第 6 章	LDAP 技术	82
6.1	目录服务与 LDAP 概述	82
6.1.1	目录服务简介	82
6.1.2	X.500 协议简介	83
6.1.3	LDAP 协议简介	84
6.1.4	LDAP 模型简介	85
6.1.5	LDAP Schema	88
6.1.6	LDAP 认证方式	91
6.1.7	LDIF 数据交换文件	94
6.2	常见 LDAP 产品介绍	97
6.2.1	IBM TDS	97
6.2.2	Sun Java 系统目录服务器	97
6.2.3	Novell eDirectory	98
6.2.4	GBase 8d	98
6.2.5	OpenLDAP	99
6.2.6	Microsoft Active Directory	99
6.3	LDAP 部署与优化	100
6.3.1	复制介绍	100
6.3.2	引用机制介绍	101
6.3.3	复制机制的部署	102
6.3.4	引用机制的部署	104
6.3.5	LDAP 优化	105
6.4	面向 LDAP 的系统设计与开发	106
6.4.1	LDAP 管理工具	106
6.4.2	应用接口编程与实例	109
6.4.3	LDAP 应用案例	119
第 7 章	实验一	120
7.1	DER 编码示例: X.501 Name 类型	120
7.1.1	ASN.1 描述与实例	120
7.1.2	DER 编码过程	121

7.2	RSA 算法示例	123
7.2.1	密钥产生	123
7.2.2	加密解密	124

第三部分 PKI 之数字证书与私钥：网络身份证

第 8 章	公/私钥格式	126
8.1	RSA	126
8.2	SM2	128
第 9 章	数字证书格式	130
9.1	基本格式	130
9.1.1	证书域组成 (Certificate)	130
9.1.2	证书内容 (tbsCertificate)	130
9.2	标准扩展项	135
9.2.1	标准扩展项 (Standard Extensions)	135
9.2.2	专用互联网扩展项	145
9.3	国内扩展项	146
9.3.1	卫生系统专用扩展项	146
9.3.2	国内通用扩展项	147
第 10 章	数字证书分类	150
10.1	根据证书持有者分类	150
10.2	根据密钥分类	150
第 11 章	私钥与证书存储方式	152
11.1	证书保存形式	152
11.1.1	DER 文件形式	152
11.1.2	Base64 文件形式	154
11.1.3	PKCS#7 文件形式	154
11.1.4	Windows 证书库形式	155
11.2	私钥保存形式	157
11.2.1	PKCS#8 文件形式	158
11.2.2	PKCS#12 文件形式	158
11.2.3	Java Keystore 文件形式	160
11.2.4	密码设备形式	161
11.2.5	软件系统形式	162
第 12 章	私钥与证书访问方式	164
12.1	CryptoAPI	164

12.1.1	CryptoAPI 简介	164
12.1.2	使用证书	166
12.1.3	使用私钥	168
12.2	PKCS#11	172
12.2.1	PKCS#11 简介	172
12.2.2	使用证书	178
12.2.3	使用私钥	181
12.3	JCA/JCE	183
12.3.1	JCA/JCE 简介	183
12.3.2	使用证书	187
12.3.3	使用私钥	189
12.4	CNG	190
12.4.1	CNG 简介	190
12.4.2	使用证书	195
12.4.3	使用私钥	196
12.5	PC/SC	200
12.5.1	PC/SC 简介	200
12.5.2	使用证书	202
12.5.3	使用私钥	213
12.6	国密接口	213
12.6.1	国密接口简介	213
12.6.2	使用证书	215
12.6.3	使用私钥	217
第 13 章	实验二	222
13.1	RSA 公钥格式编码示例	222
13.1.1	ASN.1 描述与实例	222
13.1.2	DER 编码过程	222
13.2	数字证书格式编码示例	223
13.2.1	ASN.1 描述与实例	223
13.2.2	DER 编码过程	225
13.3	Windows 证书库操作示例	229
13.3.1	查看证书库内容	229
13.3.2	导入证书	230
13.3.3	导出证书	233
 第四部分 PKI 之 CA 与 KMC: 管理网络身份证		
第 14 章	系统结构	236
14.1	国际标准	236

14.2 国内标准	237
14.2.1 证书认证系统 CA	237
14.2.2 密钥管理系统 KMC	239
第 15 章 系统设计	241
15.1 证书认证系统 CA	241
15.1.1 用户注册管理系统 RA	241
15.1.2 证书/CRL 签发系统	242
15.1.3 证书/CRL 存储发布系统	243
15.1.4 证书/CRL 查询系统	244
15.1.5 证书管理系统	245
15.1.6 安全管理系统	245
15.2 密钥管理系统 KMC	246
15.3 企业级 CA 总体设计示例	248
15.3.1 技术路线选择	248
15.3.2 模块设计	250
15.3.3 数据库设计	251
15.3.4 双证书技术流程设计	253
第 16 章 对外在线服务	256
16.1 OCSP/SOCSP 服务	256
16.1.1 OCSP	256
16.1.2 SOCSP	258
16.2 CRL 服务	259
16.2.1 基本域组成 (CertificateList)	259
16.2.2 CRL 内容 (tbsCertList)	260
16.2.3 CRL 扩展项 crlExtensions	262
16.2.4 CRL 条目扩展项 crlEntryExtensions	265
16.3 LDAP 服务	267
16.3.1 发布数字证书到 LDAP	267
16.3.2 访问 LDAP 获取数字证书	268
第 17 章 网络部署结构	270
17.1 运营型 CA	270
17.2 企业级 CA	273
17.2.1 双层标准模式	273
17.2.2 双层简化模式	273
17.2.3 单层单机模式	274
17.2.4 纯硬件模式	274

17.3 按企业管理模式部署 CA	276
17.3.1 单机构	276
17.3.2 集团公司+集中部署+集中发证	276
17.3.3 集团公司+集中部署+分布发证	277
17.3.4 集团公司+两级部署+分布发证	278
第 18 章 实验三	280
18.1 OpenSSL CA 示例	280
18.1.1 简介	280
18.1.2 安装配置	280
18.1.3 申请证书	284
18.1.4 生成并下载 CRL	287
18.1.5 导入 CA 证书到 IE 可信任证书库	290
18.2 EJBCA 示例	291
18.2.1 简介	291
18.2.2 安装配置	292
18.2.3 申请证书	300
18.2.4 下载 CRL	303
第五部分 PKI 之应用：使用网络身份证	
第 19 章 基本应用	308
19.1 身份认证	308
19.2 保密性	310
19.3 完整性	311
19.4 抗抵赖性	312
19.5 证书有效性验证	314
第 20 章 通用应用技术	315
20.1 SSL/TLS (Secure Socket layer/Transport Layer Security)	315
20.1.1 概述	315
20.1.2 记录协议	315
20.1.3 握手协议	316
20.1.4 警告协议	317
20.1.5 改变密码约定协议	318
20.1.6 应用数据协议	318
20.2 IPSec	318
20.3 Kerberos	323
20.4 TSP	326

20.5	SET	331
20.6	3-D Secure	333
20.7	WAP	335
20.8	S/MIMI	338
第 21 章	常见应用	345
21.1	防止假网站与 Web 服务器证书	345
21.1.1	假网站	345
21.1.2	使用 Web 服务器证书预防假网站	346
21.2	防止假软件与代码签名证书	348
21.2.1	Web 技术的发展	348
21.2.2	插件技术与假网银软件	350
21.2.3	使用代码签名证书预防假网银软件	351
21.3	网上银行系统	352
21.3.1	简介	352
21.3.2	应用安全需求	353
21.3.3	应用安全总体架构	354
21.4	网上报税系统	355
21.4.1	简介	355
21.4.2	应用安全需求	356
21.4.3	应用安全总体架构	356
21.5	电子病历系统	357
21.5.1	简介	357
21.5.2	应用安全需求	357
21.5.3	应用安全总体架构	359
21.5.4	网络部署结构	359
21.6	公交 IC 卡在线充值系统	361
21.6.1	简介	361
21.6.2	应用安全需求	361
21.6.3	应用安全总体架构	362
21.6.4	充值交易流程	362
第 22 章	实验四	365
22.1	Windows IIS 服务器证书配置	365
22.1.1	下载并安装服务器证书	366
22.1.2	配置 SSL 策略	373
22.1.3	访问 Web Server	374
22.2	Apache 服务器证书配置	375
22.2.1	下载并安装服务器证书	375

22.2.2	配置 SSL 策略	379
22.2.3	访问 Web Server	381
22.3	Tomcat 服务器证书配置	381
22.3.1	下载并安装服务器证书	381
22.3.2	配置 SSL 策略	384
22.3.3	访问 Web Server	384

第六部分 PKI 之运营：CA 中心

第 23 章	机房建设	388
23.1	业务系统	388
23.1.1	证书认证中心	388
23.1.2	密钥管理中心	390
23.2	应用安全	391
23.3	数据备份	394
23.4	系统可靠性	394
23.5	物理安全	394
23.6	人事管理制度	396
第 24 章	运营文件	397
24.1	CPS	397
24.2	CP	398
24.3	RA 管理	399
第 25 章	业务管理	402
25.1	管理模式	402
25.1.1	总体框架	402
25.1.2	具体要求	404
25.1.3	管理模式示例	410
25.2	主要业务流程	412
25.2.1	证书申请类	412
25.2.2	证书作废类	417
25.2.3	证书查询类	418
25.3	客户服务	420
第 26 章	资质申请	422
26.1	电子认证服务使用密码许可证	422
26.1.1	政策法规要点	422
26.1.2	申请流程	423
26.2	电子认证服务许可证	424

26.2.1 政策法规要点	424
26.2.2 申请流程	425
26.3 电子政务电子认证服务管理	426
26.4 卫生系统电子认证服务管理	427
26.4.1 政策法规要点	427
26.4.2 接入流程	428

第七部分 PKI 之法规与标准

第 27 章 国内法规	432
27.1 电子签名法	432
27.2 电子认证服务管理办法	436
27.3 电子认证服务密码管理办法	440
27.4 电子政务电子认证服务管理办法	443
27.5 卫生系统电子认证服务管理办法	447
27.6 商用密码管理条例	449
27.7 商用密码科研管理规定	452
27.8 商用密码产品生产管理规定	454
27.9 商用密码产品销售管理规定	456
27.10 商用密码产品使用管理规定	458
27.11 境外组织和个人在华使用密码产品管理办法	459
第 28 章 国内标准	461
28.1 通用性标准	461
28.1.1 祖冲之序列密码算法 (GM/T 0001)	461
28.1.2 SM4 分组密码算法 (GM/T 0002)	461
28.1.3 SM2 椭圆曲线公钥密码算法 (GM/T 0003)	461
28.1.4 SM3 密码杂凑算法 (GM/T 0004)	462
28.1.5 SM2 密码算法使用规范 (GM/T 0009)	462
28.1.6 SM2 密码算法加密签名消息语法规则 (GM/T 0010)	463
28.1.7 数字证书认证系统密码协议规范 (GM/T 0014)	463
28.1.8 基于 SM2 密码算法的数字证书格式规范 (GM/T 0015)	464
28.1.9 通用密码服务接口规范 (GM/T 0019)	464
28.1.10 证书应用综合服务接口规范 (GM/T 0020)	467
28.1.11 IPsec VPN 技术规范 (GM/T 0022)	467
28.1.12 SSL VPN 技术规范 (GM/T 0024)	468
28.1.13 安全认证网关产品规范 (GM/T 0026)	468
28.1.14 签名验签服务器技术规范 (GM/T 0029)	469
28.1.15 安全电子签章密码技术规范 (GM/T 0031)	469

28.1.16	时间戳接口规范 (GM/T 0033)	470
28.1.17	基于 SM2 密码算法的证书认证系统密码及其相关安全技术规范 (GM/T 0034)	470
28.1.18	证书认证系统检测规范 (GM/T 0037)	471
28.1.19	证书认证密钥管理系统检测规范 (GM/T 0038)	472
28.1.20	证书认证系统密码及其相关安全技术规范 (GB/T 25056)	473
28.1.21	电子认证服务机构运营管理规范 (GB/T 28447)	473
28.2	行业性标准	474
28.2.1	卫生系统电子认证服务规范	474
28.2.2	卫生系统数字证书应用集成规范	475
28.2.3	卫生系统数字证书格式规范	475
28.2.4	卫生系统数字证书介质技术规范	476
28.2.5	卫生系统数字证书服务管理平台接入规范	477
28.2.6	网上银行系统信息安全通用规范 (JR/T 0068)	477
第 29 章	国际标准	479
29.1	PKCS 系列	479
29.2	ISO 7816 系列	491
29.3	IETF RFC 系列	494
29.4	Microsoft 规范	502
29.5	Java 安全 API 规范	504
29.6	CCID 规范	506
附录	主要参考资料	507

第一部分

如何理解 PKI

第1章 为什么会出现 PKI 技术

1.1 保密通信催生了密码技术

从古到今，军队历来是使用密码技术最频繁的地方，因为保护己方秘密并洞悉敌方秘密是克敌制胜的重要条件。正如《孙子兵法》中所说：“知己知彼，百战不殆；不知彼而知己，一胜一负；不知彼不知己，每战必败。”

1.1.1 古代中国军队的保密通信方法

明末清初著名的军事理论家揭暄所著《兵经百言》系统阐述了中国古代军队的通信方法：军队分开行动后，如相互之间不能通信，就要打败仗；如果能通信但不保密，则也要被敌人暗算。所以除了用锣鼓、旌旗、骑马送信、燃火、烽烟等联系外，两军相遇，还要对暗号。当军队分开有千里之远时，宜用机密信进行通信。机密信分为三种：改变字的通常书写或阅读方式；隐写术；不是把书信写在常用的纸上，而是写在特殊的、不引人注意的载体上。这些通信方式连送信的使者都不知道信中的内容，但收信人却可以接收到信息。

1.1.1.1 阴符和阴书

公元前 1000 多年前，西周开国功臣姜子牙所著《六韬》中，讲述了战争中君主与在外将领保密通信的两种方法：阴符和阴书。

阴符共有八种：一种长一尺，表示大获全胜，摧毁敌人；一种长九寸，表示攻破敌军，杀敌主将；一种长八寸，表示守城的敌人已投降，我军已占领该城；一种长七寸，表示敌军已败退，远传捷报；一种长六寸，表示我军将誓死坚守城邑；一种长五寸，表示请拨运军粮，增派援军；一种长四寸，表示军队战败，主将阵亡；一种长三寸，表示战事失利，全军伤亡惨重。如奉命传递阴符的使者延误传递，则处死；如阴符的秘密被泄露，则无论无意泄露者或有意传告者也处死。只有国君和主将知道这八种阴符的秘密。这就是不会泄露朝廷和军队之间相互联系内容的秘密通信语言。

阴书是一种特殊书信，用于君主和主将之间军机大事的秘密联络。阴书都要拆分成三部分，并分派三人发出，每人拿一部分。只有这三部分合在一起才能读懂信的内容。

1.1.1.2 虎符、信牌和字验

古代中国的君王常以虎符作为调用军队的凭证。虎符一般由铜、银等金属制成，背面刻有铭文，以示级别、身份、调用军队的对象和范围等；虎符分为两半，一半放在朝廷，另一半由在外的将帅保管。朝廷派来的使者，需携虎符验合，才可调兵遣将。春秋战国时期，魏信陵君使如姬窃取魏王的虎符，并以此夺取大将晋鄙的兵权，然后率兵大破秦军，解了赵国之围。

公元 1040 年左右,北宋仁宗时期兵书《武经总要》中,讲述了三种军队中秘密通信的方法:符契、信牌和字验。

符契是《六韬》中阴符的改进。其中的“符”是皇帝派人向军队调兵的凭证,共有 5 种符,各种符的组合表示调用兵力的多少,每符分左右两段,右段留京师,左段由各路军队的主将收掌。使者将带着皇帝的命令和由枢密院封印的相应的右符,前往军队调兵;主将听完使者宣读皇帝的命令后,须启封使者带来的右符,并与所藏的左符验合,才能接受命令;然后用本将军的印重封右符,交由使者带回京师。“契”是主将派人向镇守各方的下属调兵的凭证,共有三种契,每契都是鱼形,可分为上下两段。上段留主将收掌,下段交各处下属收掌,使用方法类似于符。

信牌是两军阵前交战时,派人传送紧急命令的信物和文件。北宋初期使用的信物是一分两半的铜钱,后来改用木牌,上面可以写字。

字验则是秘密传送军情的一套方法。先约定 40 种不同的军情,然后用一首含有 40 个不同字的诗,令其中每一个字对应一种军情。传送军情时,写一封普通的书信或文件,在其中的关键字旁加印记。军使在送信途中,不怕被敌方截获并破解信中内容。将军们收到信后,找出其中加印记的关键字,然后根据约定的 40 字诗来查出该字所告知的情况,还可以在这些字上再加印记,以表示对有关情况的处理,并令军使带回。

1.1.1.3 矾书

矾书是用明矾水写的书信。当水干后,纸上毫无字迹,把纸弄湿后,字迹重新显现。

据记载,矾书是中国古代军事和政治斗争中常用的秘密通信方法。南宋李心传(1166—1243)所撰《建炎以来繫年要录·建炎元年正月》、元代《金史·宣宗纪上》中均记录有用明矾写的机密信。清康熙五十四年(1715 年)发生过“矾书事件”。

1.1.2 传统密码学与古代西方保密通信方法

1.1.2.1 传统密码学

西方国家大都使用拼音文字,只有二十几个字母和几个标点符号,文字符号较少,所以很适合对文字内容进行各种变换以实现保密通信。同时,由于西方世界率先进行了工业革命,发明了机器和电报等先进技术,所以,西方一直在引领着密码学的发展,尤其是近代以来。因此,逐步形成了以西方拼音文字为主要研究和操作对象的传统密码学理论和方法,这些理论和方法并不完全适用于中国的象形表意文字。

传统密码学主要分为两大类:换位加密法和替换加密法。

1. 换位加密法

换位加密法是指在加密过程中不改变字或字母本身,只改变它们的排列顺序,以达到加密的目的。古希腊军队使用的 *Scytale* 棒就是一种简单的换位加密法。

另一种常用且较复杂的换位加密方法是按一种特定的路径,把信文写在一张信纸上,然后在信纸的其他空白处写上无关紧要的文字,信件按正常顺序读起来就是一封普通的信。收信者只需按照规定好的路径去读,就能获取真正的信文。中国古代的字验就采用这种换位加密方法。

中国古代的“藏头诗”也是一种换位加密方法：将每句诗的第一个字连起来读，就会表达另一种含义。如《水浒传》中梁山为了拉卢俊义入伙，智多星吴用在玉麒麟卢俊义家中的墙上写下一首藏头诗：“芦花滩上有扁舟，俊杰黄昏独自游。义到尽头原是命，反躬逃难必无忧。”该诗巧妙地把“卢俊义反”四个字暗藏于四句之首，而一心躲避“血光之灾”的卢俊义并没有细察这其中的隐秘。果然，这四句诗写出后，被官府拿到了证据，大兴问罪之师，到处捉拿卢俊义，终于把他逼上梁山。

意大利文艺复兴时期的名人达芬奇，总是用“镜像法”记笔记，据说是为了防止别人偷窥他的秘密。也就是说，他的笔记本只有通过镜子才能被正常阅读。因此，“镜像法”也是一种换位加密法。

2. 替换加密法

替换加密法是指不改变信文中字或字母的顺序，而是用其他的符号来替换它们，以达到加密的目的。根据信文中每个字（字母）使用一个或多个字（字母）替换，又分为单表替换和多表替换。

古罗马军队的凯撒（Caesar）加密就属于单表替换加密，此类替换加密很容易被破解。

维吉尼亚（Vigenere）密码和杰弗逊（Jefferson）加密器属于多表替换加密，其破解难度要比单表替换加密大得多。近代以来，西方主要使用的是各种多表替换加密方法，甚至在第二次世界大战中使用的那些极其复杂的机器密码，本质上也是多表替换加密。

1.1.2.2 古希腊军队的 Scytale 棒

大约公元前 700 年，古希腊军队使用 Scytale 棒的加密方法是一种简单的换位加密法。其使用方法是：把长带子状羊皮纸缠绕在圆木棍上，然后在上面写字；解下羊皮纸后，上面只有杂乱无章的字符，只有再次以同样的方式缠绕到同样粗细的棍子上，才能看出所写的内容。例如，在缠绕于木棍上的纸带上写下英文字：tommorrow midnight attack，表示开始军事进攻的时间；然后把纸带从木棍上解下来，英文字将变成：tmaoitmdtmnaoicrgkrhow。

这种 Scytale 圆木棍也许是人类最早使用的文字加密解密工具，据说主要是古希腊城邦中的斯巴达人（Sparta）在使用它，所以又被称作“斯巴达棒”。

1.1.2.3 古罗马军队的凯撒（Caesar）加密

公元前 100 年前，古罗马军队统帅凯撒设计了一种军事信息加密方法，用于同手下的将军们进行保密通信。所有字母用字母表中顺序循环移位后的字母替换。当规定字母表顺序移动 3 位时，则 a 替换为 d，b 替换为 e，c 替换为 f，……，x 替换为 a，y 替换为 b，z 替换为 c，军事信息 tommorrow midnight attack 就变成 wrppruurz plgqljkw dwwdfn。如果不知道加密方法，就无法知道该信息的实际意义；解密时，只需把所有字母逆序移动 3 位，就能获得军事信息明文。

凯撒密码属于单表替换法，而且替换的规则很简单。

1.1.2.4 中世纪后保密通信方法

13 世纪开始，欧洲经历了一场提倡人文精神和思想解放的影响深远的文艺复兴运动，这一时期，欧洲诸国开始进行现代意义上的外交活动，纷纷在别国建立大使馆并互派大使。

身处异国的大使要向自己的政府汇报高度机密的信息，却只能通过两国之间的公共邮政系统来传递，为此外交官们不得不绞尽脑汁使用各种复杂的加密方法来写信。另外，情报机关也在费尽心机破解这些信。各方斗智的结果便促使了密码技术的迅速发展。

16 世纪开始，欧洲又掀起了一场范围广泛的基督教改革运动，因为涉及敏感的教义等问题，极其活跃的宗教界人士都在用五花八门的加密方法写信交流。

欧洲诸国的皇宫内也布满了各种阴谋诡计，密码当然成为不可缺少的酝酿阴谋的工具，同时，破解密码也成为揭露阴谋的最有效手段。种种因素导致密码在欧洲的外交界、宗教界和政界大肆流行，传统密码学迎来了它的繁荣时期。

1. 维吉尼亚（Vigenere）密码

由于单表替换加密很容易被频率分析法破解，15 世纪中叶开始，欧洲人开始研究各种“多表替换加密”方法，即信文中同一个字母在不同的位置会有不同的替换符号，其中最有名的是“维吉尼亚密码”。

维吉尼亚密码因为其原理简单、使用方便而广受欢迎，它使用一张字母表矩阵：第一行是任意给定的字母替换表，第二行是第一行表顺移一位而形成的字母替换表，第三行表又是第二行表的顺移一位，以下各行以此类推。加密时，对于信文中的同一个字母，当其第一次出现时使用表的第一行替换，第二次出现时使用第二行替换，以此类推。如果该字母出现次数已超过矩阵的行数，则回到第一行继续。解密同加密一样，也是从上到下逐行进行。

显然，多表加密比单表加密复杂许多，因此其破解难度也加大许多。自从维吉尼亚密码出现以后，多表加密成为欧洲人主要使用的加密方法。

2. 欧洲的黑室

到了 18 世纪，欧洲各国相继成立了专门截获和破解外交邮件中机密信息的机构，这些机构被称为“黑室”，其存在及所作所为在当时是各国之间心照不宣的秘密。黑室中的密码专家经常因破译重要的机密信而获得皇室的嘉奖。

3. 杰弗逊（Jefferson）加密器

美国第三任总统托马斯·杰弗逊于 1795 年发明了一种加密装置叫杰弗逊转轮加密器。该装置有 36 片同样大小的木制转轮，套在一根铁杆上。每片转轮的圆周边缘上刻有乱序的 26 个英文字母。其使用方法是：进行秘密通信的双方必须各自拥有完全一样的转轮加密器，当一方要把一段文字（不超过 36 字）秘密通知身处异地的对方时，只需转动加密器上的各片转轮，使这段文字正好出现在同一行上，这时转轮上排列的其他 25 行都是无意义的乱码；再把其中任意一行的乱码抄下来交给信使。信使并不知道这段乱码文字的意义，只负责把它送交对方。对方收到乱码信后，只需拿出自己的转轮加密器，转动上面各片转轮，让其中一行的排列和这段乱码同处在一行上，然后再查看其他 25 行上的内容，其中必然有一行显示出加密者要传达的信息，而其他行显示的都是乱码。

杰弗逊加密器属于多表替换加密（每一个转轮相当于一张替换字母表），它很难被破解，除非得到通信双方所使用的加密装置。据称，美国军队到了 20 世纪 60 年代仍在使用它。轮转式加密器在第二次世界大战时很流行，不过都是机械电子式的，并且与打字机等设备结合使用，比杰弗逊的最初原始装置高级很多。

1.1.2.5 古代阿拉伯人开创密码分析学

公元 9 世纪，阿拉伯数学家、哲学家肯迪所著《解码手册》是历史上最早研究用频率分

字母	频率	字母	频率
a	0.08167	n	0.06749
b	0.01492	o	0.07507
c	0.02782	p	0.01929
d	0.04253	q	0.000009
e	0.12702	r	0.05981
f	0.02228	s	0.06327
g	0.02015	t	0.09056
:	:	:	:

图 1-1 英文字母频率表

析法破解密码的文献。英文文章中每个字母出现的频率是不同的，如图 1-1 所示。由于 E 频率最高，如果一份密文中 R 出现的最多，则 R 可能就是 E；即使不是 E，也应是明文中出现频率较高的字母。按照这种频率分析方法，密码就容易破解。

基于字母和单词统计学特征的频率分析方法一直是破解密码最基本和最常用的方法。两次世界大战中的密码战，是当时敌对双方最优秀的科学大脑和最先进的科技之间的生死较量，但究其所依据的加密和破解原理，仍然是基于字母和单词的频率分析，只是复杂程度更高而已。

1.1.3 两次世界大战的密码斗法

1.1.3.1 第一次世界大战

1837 年，美国人发明了电报，1896 年前后，意大利人和俄罗斯人分别发明了无线电报，从此人类进入了电子通信时代。有线和无线电报能快速方便地进行远距离收发，因此很快成为军事上的主要通信手段。但是，有线电报需经过电缆传送，而设在海底和野外的电缆很容易被破坏或窃听；而无线电报是一种广播式通信，包括敌人在内的任何人都能够接收到发射在空中的电报信号。因此，为了防止机密泄露，电文的加密变得至关重要。

1914 年 6 月 28 日，塞尔维亚一位 19 岁青年人在其首都萨拉热窝，刺杀主张吞并塞尔维亚的奥匈帝国王储夫妇，由此引发了第一次世界大战，以英、俄、法为首的协约国对抗以德、奥匈帝国、奥斯曼土耳其为首的同盟国。

交战各方相继成立了专门的密码机构，密码专家们开始斗法。争斗中，大家互有胜负：德军截获到俄军的无线电通信，洞悉了其军事部署，结果把拥有优势兵力的俄国人打得大败；不久，战败的俄国在国内爆发了十月革命；法国人则数次破译了德军的密码，成功粉碎了德军攻占巴黎的阴谋。

然而，这场密码战中最大的赢家似乎是英国。1914 年 8 月 4 日，英国向德国宣战，当天就割断了德国的所有海底电缆，迫使德军只能严重依赖无线电报和国外电缆进行通信联系，使得英国海军部很容易地截获了大量的德国电报情报信号。为破解这些加密情报，英国海军部于 1914 年 10 月成立了一个代号为“40 号房间”的密码破译小组，在第一次世界大战期间，该小组成功破译了约 15 000 份德国密电，使得英国海军在与德国海军的交战中屡次占得先机，并最终帮助协约国赢得了战争。

1917 年 4 月 6 日，美国终于向德国宣战。美国陆军部于 1917 年 6 月成立军事情报处第八科，也被称为“美国黑室”。该机构成立时只有 3 人，一年后就发展成近 200 人的庞大组织，下设 5 个部门：密码编写组、通信组、速记组、密写组和密码破译组。该机构破译了德国和日本的大量密码，帮助美国赢得了军事和政治上的胜利。1929 年 10 月，“美国黑室”被新任美国国务卿下令关闭。自 1917 年成立到 1929 年关闭，该机构解读了 45 000 多份密码电

报，破解了包括中国、德国、英国、法国、俄国、日本等在内的 20 多个国家的密码。

1918 年 11 月 11 日，协约国和同盟国宣布停火，并经长达 6 个月的谈判后，于 1919 年 6 月 28 日在巴黎凡尔赛宫签署条约，标志第一次世界大战正式结束。

虽然密码的应用在第一次世界大战中大显身手，但密码学作为一门学科，在此期间并没有突破性的发展，使用的加密方法与古代相比并没有什么创新，只是增加了一些难度。

1.1.3.2 第二次世界大战

第二次世界大战中密码学的发展远远超过了以往任何时代，无论是在密码学的技术、理论还是应用方面，在此期间都发生了革命性变化。

从密码学的技术方面来看，基于机械和电气原理的加密/解密装置全面取代了以往的手工密码，不仅大大提高了加密/解密的速度，也使密码的复杂度大大提高，用传统的手工方式已不可能破解这些机器密码。另外，“矛”的进化也促进了“盾”的发展。破解密码也实现了机械化和电气化，甚至开始使用电子计算机，使得破解密码的效率大大提高。人类从此进入了机器密码时代。

从密码学的理论方面来看，大量的数学和统计学知识被应用于密码分析和破解，并大获成功。于是，越来越多的数学家不断加入密码队伍，他们开始取代语言学专家、象棋冠军和猜谜高手，成为密码战场上的主力军。

从密码学的应用方面来看，参战各国都已认识到密码决定着战争胜负的关键。于是，各国都纷纷研制和采用最先进的密码设备，建立最严格的密码安全体系。如德国军队全面使用“隐谜”密码打字机，德军最高司令部使用“洛伦兹”密码电机，日本外交官使用“紫色”密码打字机，日本海军使用 JN 系列密码等。

另外，各国都投入大量的人力和物力，集中最优秀的密码专家们想方设法破解敌国的密码。如英国和美国都拥有上万人的密码队伍，专门从事破解德国和日本的军事和外交密码的工作。

英美政府和军队把通过破解密码所获得的情报称为“超级情报”，其重要性甚至超过传统的“顶级密码”。他们竭力保护“超级情报”的来源，不允许它们有任何泄露。正是依赖这些“超级密码”，才使得他们能扭转战争中的被动局面，在战场上取得一系列重要的胜利，最终大败了法西斯强敌。

在中国的抗日战场上，国民党政府建立的“军委技术研究室”也成功破解了日本空军的密码及日本外务省的部分密码，并且与美英两国的密码专家和情报机构建立了合作关系。

1.1.4 现代密码学与信息时代

第二次世界大战时期的密码学经历了历史上前所未有的一场革命，这场革命几乎颠覆了传统密码学中所有的理论和方法，从而迎来了机器密码的时代。这个时代的一个典型代表就是德国的“隐谜”密码机与波兰、英国和美国的“炸弹”破译机这两种机械密码装置之间惊心动魄的较量。

然而谁会想到，曾经如此辉煌的机器密码时代，竟然在第二次世界大战结束后不久，就很快终结了。因为从 20 世纪 50 年代开始，计算机技术突飞猛进，在具有超强计算能力的计算机面前，所有的机器密码都显得不堪一击。

此外, 由于美国通信专家 Shannon 于 1949 年创立了信息论, 人们知道的所有文本、声音、图像、视频信息都能够转换成数字形式, 从而可以用功能强大的计算机来处理。到了 20 世纪 70 年代, 美国政府已经拥有大量数字化的计算机文件, 如何保证这些机密的计算机文件不被偷看、窃取和篡改, 成为计算机时代下密码学的一个新任务。

与此同时, 电子通信技术也在计算机的支持下迅猛发展。继电话和电报之后, 又出现了计算机通信网络。这种通信网络很快遍布全球的每个角落, 从而把整个世界联系在一起, 从此人类进入信息时代。在信息时代如何保证计算机网络通信和数据传递的安全性, 又成为密码学的一个全新任务。

由于计算机的出现, 数字化信息的产生和网络通信的发展, 促使密码学经历了一场比第二次世界大战时期的机器密码更彻底的革命。在这场革命中出现了一种新的加密对象, 既不是几千年来文字书写, 也不是有 100 多年历史的电报字码, 而是数字化的文本。

数字化的一个直接后果是, 人们从此可以对文本和信息施以复杂的数字运算, 以实现种种控制目的, 包括加密和解密。于是, 大量的代数、组合、数论、概率统计等数学知识被用于密码学, 使得它成为数学的一个新分支。

现代密码学的任务已不只限于传统密码学的“保密通信”, 而是含义更广的“信息安全”, 其中包括“保密通信”、“数据加密”、“数字签名”等重要功能, 并且其应用也远远突破了军事、外交和捷报等传统的范围, 开始全面进入经济、商务、科学、教育等人类社会活动的各个领域, 从而给我们的工作和生活带来深远的影响。现代密码学已经成为信息时代无处不在且不可缺少的信息安全卫士。

与传统密码学不同, 现代密码学设计时, 一般总是假设密码系统的结构是公开的, 或者至少为敌人所知。这一假设被称为科考夫原则 (Kerckhoffs's Principle), 是由 19 世纪荷兰密码专家 Auguste Kerckhoffs 首先提出的, 该假设之所以在密码学界被普遍接受, 是因为它基本符合实际情况, 并且据此能简化密码系统的分析、设计和实施。密码系统的结构称作密码算法, 进行加密或解密操作所需要的关键参数称作密钥。事实上, 在日常社会活动领域中使用的密码算法基本上都是公开的。现代密码学的安全性主要取决于密钥的设计和使用。

根据技术特征, 现代密码学可分为三类。

1. 对称算法

对称算法是指加密密钥和解密密钥相同的密码算法, 又称密码密钥算法或单密钥算法。该类算法又分为流密码算法和分组密码算法。

流密码算法又称序列密码算法, 每次加密或解密一位或一字节的明文或密文。

分组密码算法将明文 (密文) 分成固定长度的数据块 (比特块或字节块), 用同一密钥和算法对每一明文 (密文) 块加密 (解密) 后得到等长的密文 (明文) 块, 然后将密文 (明文) 块按照顺序组合起来最终得到密文 (明文)。

常见的流密码算法包括 RC4; 常见的分组密码算法包括 DES、IDEA、RC2、AES、SM4 等。

2. 非对称算法

非对称算法是指加密密钥和解密密钥不相同的密码算法, 从一个密钥很难推导出另一个密钥, 又称公开密钥算法或公钥算法。该算法使用一个密钥进行加密, 用另一个密钥进行解密, 其中加密密钥可以公开, 又称公开密钥或公钥; 解密密钥必须保密, 又称私有密钥或私钥。

常见的非对称算法包括 RSA、DH、DSA、ECDSA、ECC、SM2 等。

3. 摘要算法

摘要算法是指把任意长的输入消息数据转化成固定长度的输出数据的一种密码算法，又称散列函数、哈希函数或杂凑函数、单向函数等。摘要算法所产生的固定长度的输出数据称作摘要值、散列值或哈希值。摘要算法没有密钥。

常见的摘要算法包括 MD5、SHA1、SM3 等。

1.2 密码技术普及推动了密钥管理技术的发展

1.2.1 密钥管理

在密码算法公开的情况下，现代密码学的安全性主要取决于密钥的安全性。如果获得对方当前使用的密钥，密码破译者就可很轻易地从截获的密文中破解出明文来。由于现代密码学依据科考夫公开原则，算法的设计运用了多种数学知识，并经过了大量的学术研究，是很多人的智慧结晶，因此算法自身的安全性很高。密码破译者通过分析算法可能存在的安全漏洞或缺陷，来推测或演算出对方当前使用的密钥，这种方式难度很大，成本也很高。

聪明的密码破译者开始通过其他方式来获取对方当前使用的密钥。如果密钥使用生日、姓名、单词等符号组成，通过多次尝试就可被猜测出来。如果密钥存储不安全，就可以从对方计算机硬盘或内存中获得密钥。也可以在对方传递密钥过程中，截获或者替换密钥。还可以直接贿赂或收买关键人物，直接获得密钥。各种实例说明，从人身上找到漏洞比找到密码系统的漏洞更容易。

于是，密钥管理应运而生，通过对密钥全生命周期进行安全管理，从而实现密钥的安全性。密钥管理主要包括：密钥产生、密钥传输、密钥验证、密钥更新、密钥存储、密钥备份、密钥销毁、密钥有效期、密钥使用等。广义的密钥管理不仅包括密钥如何分发到用户，也包括密钥如何使用。狭义的密钥管理并不包括用户获得密钥后如何使用。

1. 密钥产生

目前针对密钥有两种常见的攻击方法：穷举攻击法和字典攻击法。

穷举攻击法就是通过尝试所有可能的密钥来寻找当前使用的密钥。如果使用一台每秒尝试 100 万次密钥的计算机，只需 36 分钟就可以把由小写字母和数字组成的 6 位长的所有密码尝试一遍。2013 年 9 月，世界上最快的计算机每秒运算速度最快可达 5.49 亿亿次，而且计算机的计算能力几乎每 18 个月就增加一倍。

字典攻击法就是把人们最可能使用的所有密码汇聚成一本密码字典，攻击时只尝试该字典中的所有密码。字典攻击法常用于攻击系统口令，是一种特殊的穷举攻击法。计算机上 40% 的口令可以通过字典攻击法破译。

为了抵御穷举攻击法和字典攻击法，密钥长度应足够长，密钥复杂度应足够高，同时应避免出现弱密钥。一般来说，密钥长度越大，对应的密钥空间就越大，攻击者使用穷举猜测密码的难度就越大。由自动处理设备生成的随机比特串是好密钥。对公钥密码算法来说，密钥产生更加困难，因为密钥必须满足某些数学特征。

2. 密钥传输

甲乙双方需要共享一个相同密钥才能进行保密通信。通常由一方先产生一个密钥，然后通过安全方式传递给另一方。

密钥传输方式有多种形式。可以选择人工面对面方式，也可以选择邮寄或快递方式，将密钥副本交给对方。也可以将密钥分成许多不同的部分，然后采用不同方式发送出去：有的通过电话、有的通过快递等。

X9.17 标准描述了两种密钥：密钥加密密钥和数据密钥。密钥加密密钥加密其他需要分发的密钥；而数据密钥只对信息流进行加密。密钥加密密钥一般通过手工分发。

3. 密钥验证

有时密钥在传输中会发生错误。可以通过密钥后面附着一些检错和纠错位，来检测这种错误，如果发生错误，可要求重传密钥。

甲收到乙的密钥后，如何验证该密钥是乙的密钥，而不是丙的密钥呢？一种常用的验证方法是，乙选择一段内容用该密钥加密，然后发给甲，甲解密后如果明文正确，则可以确认该密钥是乙的密钥。

4. 密钥更新

当密钥需要改变时，如何方便地获得新密钥并不是一件容易的事。可以基于旧密钥产生新密钥。如果双方事先共享同一密钥并约定相同的计算方法（如摘要算法），每次密钥更新时对共享密钥或旧密钥计算一次，就可得到新密钥。

5. 密钥存储

密钥可以记忆在大脑中，也可以直接存储到计算机硬盘、智能卡，还可以把密钥分成多个部分，分别存储到不同位置，另外还可以采用其他密钥进行加密保存。

6. 密钥备份

密钥备份可以采用密钥托管、密钥分割、密钥共享等方式。

密钥托管是把密钥交给第三方中心进行保管（如锁在保险柜里或用主密钥加密保存），一旦该密钥丢失（如遗忘密钥或用户意外死亡），按照一定的规章制度，可从该中心索取或恢复该密钥。

密钥分割是把密钥分割成许多碎片，每一碎片本身并不代表什么，但把这些碎片放到一块，就可合成该密钥。

密钥共享是将密钥分成 n 块，知道任意 m ($m < n$) 或更多块就能够计算出该密钥，但知道任意 $m-1$ 或更少的块都无法计算出该密钥，该方法又称作 (m, n) 门限（阈值）方案。

7. 密钥销毁

如果密钥必须替换，旧密钥就必须销毁。旧密钥是有价值的，即使不再使用，有了它们，攻击者就能读到由它加密的一些旧信息。

密钥必须安全地销毁。如果密钥写在纸上，这张纸必须切碎或烧掉。如果密钥存储在硬盘或内存中，存储位置必须使用其他数据多次重写。由于密钥在计算机中很容易被复制并存储到多个地方，应编写特殊的删除程序，查看所有硬盘或内存，寻找可能出现的密钥副本，

彻底删除干净；同时需要记住彻底删除所有临时文件或交换文件的内容。

8. 密钥有效期

加密密钥不能无限期使用，主要原因包括：密钥使用时间越长，它泄露的机会就越大；如果密钥已泄露，那么密钥使用越久，损失就越大；密钥使用越久，人们花费精力破译它的诱惑力就越大；对用同一密钥加密的多个密文进行密码分析一般比较容易。

对任何密码应用，必须有一个策略能够检测密钥的有效期。不同密钥应有不同的有效期。密钥有效期主要依赖数据的价值和给定时间里加密数据的数量。数据价值越大，加密数据数量越多，所用密钥的有效期就越短，更换越频繁。

9. 密钥使用

安全获得密钥后，就可以使用密钥进行保密通信了。进行保密通信时，可以直接使用该密钥对数据进行加解密，但该方式容易造成密钥泄露或容易被对方破解。为避免密钥泄露或破解风险，在进行保密通信时，并不直接使用共享密钥，而是首先基于该密钥产生会话密钥，然后再使用会话密钥对数据进行加解密。

1.2.2 对称密钥管理技术

基于数字化信息和网络通信的现代密码学，其应用范围已远远超出传统的谍报、外交和军事领域，开始向人类几乎所有的社会活动领域渗透，甚至已进入普通民众的日常生活。显然，密码应用的普及推动了密钥管理技术的发展。

19 世纪 70 年代公钥密码思想提出之前，包括传统密码学在内的所有密码技术均属于对称密码范畴，因此密钥管理技术的研究最早是从对称密钥管理入手的。对称密钥管理技术可分为两种模式，如图 1-2 所示。

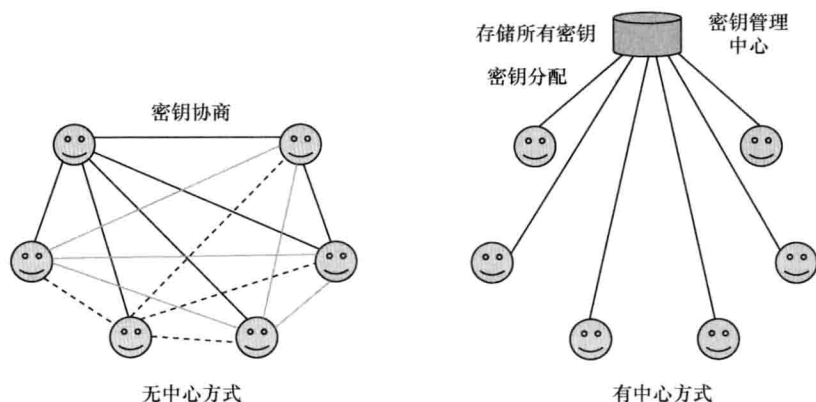


图 1-2 对称密钥管理技术的两种模式

1. 无中心模式

每对用户通过协商获得一个共享密钥。为避免两个用户之间共享密钥的安全性不受第三个用户密钥泄露的影响，每对用户之间密钥应互相独立。

当两个用户之间需要保密通信时，为避免密钥泄露，并不直接使用共享密钥对数据进行加解密，而是基于共享密钥产生会话密钥后，使用会话密钥对数据进行加解密。

无中心模式具有以下特点：

- (1) 密钥协商次数随用户数目呈指数增长。 N 个用户时，共需要密钥协商次数为 $N(N-1)/2$ 。如， $N=6$ 时为 15 次协商， $N=1000$ 时为 500 000 次协商。
- (2) 每个用户需保存与所有用户的共享密钥。 N 个用户时，每个用户需保管 $N-1$ 个密钥。
- (3) 每对用户之间密钥协商自由灵活，不受其他用户的任何影响。

由于密钥协商次数随用户数目呈指数增长，且每个用户都需要管理与所有用户的共享密钥，因此该模式只适用于小规模用户场合。

2. 有中心模式

存在一个独立的密钥管理中心，每个用户均信任该中心。密钥管理中心为每个用户分配一个密钥，并负责存储和管理所有的用户密钥。当某用户的密钥泄露后，密钥管理中心将该密钥标记为失效。

当两个用户之间需要保密通信时，需要通过密钥管理中心动态验证对方密钥的合法性：

- (1) 该密钥是否失效；(2) 该密钥是否是当前用户的。

有中心模式具有以下特点：

- (1) 密钥分配次数与用户数目呈线性关系。 N 个用户时密钥分配次数为 N ，且每个用户只需保管 1 个密钥。
- (2) 由于密钥是随机产生的，无法判断属于哪个用户，因此密钥管理中心需保存密钥与用户的映射关系，存在密钥被泄露和映射关系被篡改的安全风险。
- (3) 用户之间进行保密通信时，需要通过密钥管理中心动态验证对方密钥的合法性，既无法实现脱机保密通信，又容易导致密钥管理中心成为性能瓶颈。

由于密钥管理中心需安全存储所有密钥及其与用户的映射关系，且需在线验证密钥的合法性，因此该模式不适合于大规模的公众服务领域。

该模式目前比较成熟的应用是磁条卡密码应用体系，已经成为银行体系事实上的密码应用标准，广泛应用于几乎所有国内外银行，通过对区域（如分行、网点等）或终端（如 ATM、POS 等）分配对称密钥，实现银行卡交易过程中的数据安全。具体技术细节，请参考《商业银行密码技术应用》“第三章 磁条卡密码应用体系”。

为避免存储大量的用户密钥，同时实现脱机交易，电子钱包密码应用体系中引入“父子密钥机制”，即基于用户唯一标识（如卡号）由父密钥直接产生子密钥；在所有设备和用户端预先安全存储父密钥，无需保存任何用户密钥，当验证密钥与用户是否匹配时，通过本地存储的父密钥和用户唯一标识就可计算出该用户的密钥，从而实现脱机验证。具体技术细节，请参考《商业银行密码技术应用》“第四章 电子钱包/存折密码应用体系”。

1.2.3 非对称密码技术简化了密钥管理

由于对称密钥管理模式中加密和解密采用相同密钥，在密钥协商或分配时，容易造成密钥泄露且很难发现，直至 19 世纪 70 年代公钥密码思想提出“公钥”和“私钥”之后，才根本性地解决了这个难题。由于公钥可以公开，不存在泄露的风险，因此采用非对称密码技术可以简化密钥管理。同对称密钥管理技术类似，非对称密钥管理技术也可分为两种模式，如图 1-3 所示。

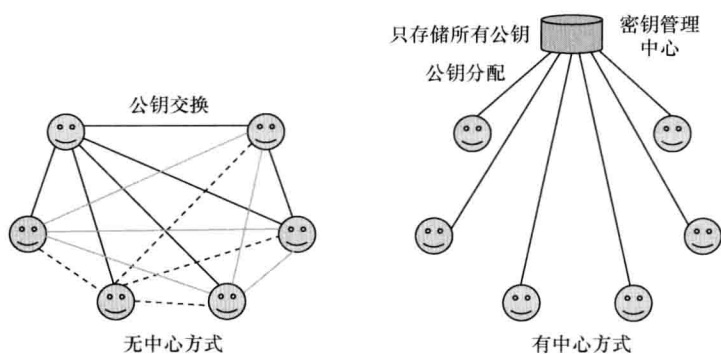


图 1-3 非对称密钥管理技术的两种模式

1. 无中心模式

每对用户通过交换获得对方公钥。当两个用户之间需要保密通信时，可以直接使用对方的公钥对数据加密，对方收到密文后用自己的私钥即可解密；也可以随机产生一个对称密钥，使用对称密钥对数据加密，再使用对方的公钥加密对称密钥，对方收到密文后，先用自己的私钥解密获得对称密钥，然后再使用该对称密钥解密数据。

无中心模式具有以下特点：

(1) 公钥交换次数随用户数目呈指数增长。 N 个用户时，共需要公钥交换次数为 $N(N-1)/2$ 。如， $N=6$ 时为 15 次交换， $N=1000$ 时为 500 000 次交换。

(2) 每个用户需保存所有用户的公钥。 N 个用户时，每个用户需保管 $N-1$ 个公钥。

由于公钥交换次数随用户数目呈指数增长，且每个用户都需要管理所有用户的公钥，因此该模式只适合于小规模用户场合。

该模式目前比较成熟的应用是 PGP 模式，在互联网上有很多独立的个人群体内部使用。

2. 有中心模式

存在一个独立的密钥管理中心，每个用户均信任该中心。密钥管理中心为每个用户分配一个公钥，并负责存储和管理所有的用户公钥。在进行密钥分配时，可由用户自己产生公钥和私钥，只把公钥提交给密钥管理中心；也可以由密钥管理中心为用户产生公钥和私钥，把公钥和私钥均安全传递给用户，但只保存公钥。当某用户的私钥泄露后，密钥管理中心将该用户的公钥标记为失效。

当两个用户之间需要保密通信时，需要通过密钥管理中心动态验证对方公钥的合法性：

(1) 该公钥是否失效；(2) 该公钥是否是当前用户的。

有中心模式具有以下特点：

(1) 公钥分配次数与用户数目呈线性关系。 N 个用户时，公钥分配次数为 N ，且每个用户只需保管 1 个私钥和公钥。

(2) 由于公钥是随机产生的，无法判断属于哪个用户，因此密钥管理中心需保存公钥与用户的映射关系，存在映射关系被篡改的安全风险。

(3) 用户之间进行保密通信时，需要通过密钥管理中心动态验证对方公钥的合法性，既无法实现脱机保密通信，又容易导致密钥管理中心成为性能瓶颈。

由于密钥管理中心需存储所有公钥与用户的映射关系，且需在线验证公钥的合法性，因

此该模式不适合于大规模的公众服务领域。

1.3 PKI 本质是把非对称密钥管理标准化

由于非对称密钥管理的无中心模式只适合用于小规模用户场合，很难适应公众普遍参与的信息时代，已经逐渐被边缘化；在信息时代，非对称密钥管理的有中心模式已经成为主流。

非对称密钥管理模式，尽管很好地解决了密钥协商或分配时密钥容易泄露的难题，但并没有解决好密钥与用户映射关系容易被篡改的问题，依然通过密钥管理中心集中存储公钥与用户的映射关系。

PKI 通过引入 CA、数字证书、LDAP、CRL、OCSP 等技术并制定相应标准，有效地解决了公钥与用户映射关系、集中服务性能瓶颈、脱机状态查询等问题；同时为促进并提高证书应用的规范性，还制定了很多与证书应用相关的各种标准。

1. CA 和数字证书

为有效解决公钥与用户映射关系容易被篡改的问题，PKI 引入“CA”和“数字证书”技术。

(1) CA 为证书权威，是 Certificate Authority 的缩写，也称作 CA 中心或证书认证中心，是一种特殊的密钥管理中心，拥有自己的公钥和私钥，负责给用户签发数字证书，即使用 CA 中心私钥对用户身份信息和公钥信息进行加密处理（签名）后形成数字证书。

(2) 数字证书是一种特殊的文件格式，包含用户身份信息、用户公钥信息和 CA 中心私钥的签名。用户身份信息包括姓名或名称、单位、城市、国家等。

当获得数字证书后，使用 CA 中心的公钥对数字证书中私钥的签名进行解密处理后，就可验证该数字证书是否被篡改，从而确认公钥与用户身份的映射关系。只要保证 CA 中心私钥的安全性，就能保证数字证书很难被篡改，从而保证了公钥与用户映射关系很难被篡改。

用户在验证数字证书是否被篡改时，必须首先获得 CA 中心的公钥。为方便用户识别 CA 中心的公钥，CA 中心也为自己签发数字证书，该证书包含 CA 中心公钥、CA 中心身份信息和 CA 中心私钥的签名。

数字证书实际上是把用户公钥、公钥与用户的绑定关系以公开形式发布出去，方便信息时代大规模用户使用。

2. LDAP

CA 中心存储着所有用户的数字证书，为方便用户快速获得交易对方的数字证书，避免 CA 中心成为性能瓶颈，PKI 又引入了 LDAP 技术，通过 LDAP 方式对外提供证书查询或下载服务。LDAP 为轻量目录访问协议，是 Light-weight Directory Access Protocol 的缩写。

专业的关系型数据库管理系统（如 Oracle、DB2 等），专门对结构化数据进行管理，应用系统只需通过 SQL 语句（报文协议）发送各种数据管理指令，而具体管理工作完全由数据库管理系统负责。尽管这种数据管理方式大大简化了应用系统的复杂度，但由于其读写功能相对均衡，很难满足高性能的查询服务。为获得高性能的查询服务，需要对数据管理的读取功能进行特殊优化，于是出现了目录服务技术（DAP）。目录服务技术对查询功能进行优化，比修改操作快 10 倍以上，适合快速响应和大容量查询服务。DAP 技术通过目录树方式对数

据进行管理，应用系统只需通过 X.500 协议就能对数据进行快速查询。

由于 X.500 协议过于复杂，实现成本很高，于是国际组织对 X.500 进行了简化，形成 LDAP 标准，而且 LDAP 支持 TCP/IP 协议，更适合于互联网领域使用。

由于数字证书是可以公开的，CA 中心也可以将其签发的数字证书存储到其他地方，供用户查询或下载。

3. CRL 和 OCSP

当用户私钥泄露后，CA 中心有责任将该用户的证书标记为失效。但用户如何获得对方证书是否失效的状态呢？为方便用户获得证书状态，PKI 引入了 CRL 和 OCSP 技术。

(1) CRL (Certificate Revocation List)。

CRL 为证书作废列表，是 Certificate Revocation List 的缩写，也称作证书黑名单，是一种特殊的文件格式，包含所有失效的证书清单、下次 CRL 生成时间和 CA 中心私钥的签名。

CA 中心定期生成 CRL，并将下次生成时间写入 CRL 中，方便用户按时定期下载 CRL。跟数字证书类似，CRL 也是通过 CA 中心私钥的签名来保证 CRL 无法被篡改的。用户只需定期获取 CRL 后，就可在本地脱机验证证书是否失效，在下次 CRL 生成时间之前无需联系 CA 中心。

由于 CRL 是可以公开的，CA 中心也可以将其签发的 CRL 存储到其他地方，供用户查询或下载。

(2) OCSP (Online Certificate Status Protocol)。

当 CA 中心将私钥已泄露的用户证书标记为失效后，如果还未到下次 CRL 生成时间，此时其他用户通过最新 CRL 并不知道该用户证书已经失效，依然将该用户当作有效用户继续进行各种保密通信和合法交易，可能会造成一定的损失。

为解决 CRL 滞后的缺陷，避免给高实时性或高风险交易造成重大损失，PKI 引入了 OCSP，对用户实时的证书状态查询服务。

OCSP 为在线证书状态协议，是 Online Certificate Status Protocol 的缩写，当用户需要实时查询对方证书是否有效时，可通过 OCSP 协议实时访问 CA 中心获得对方证书的当前状态。

4. 其他 PKI 标准

除与 CA 直接相关的上述标准外，PKI 还包括其他很多标准，如密码相关标准、证书应用标准、证书存储标准、证书访问标准、CA 运营标准等。

密码相关标准包括 ASN.1、DER、HMAC、DES、AES、RSA、ECC 等。

证书应用标准包括 SSL/TLS、SET、WAP、IPSec、TSP、PMI 等。

证书存储标准包括 PKCS 系列标准、ISO 7816 系列标准等。

证书访问标准包括 PKCS 11、CryptoAPI、JCE、PC/SC 等。

CA 运营标准包括 CP、CPS 等。

1.4 私钥专有性使人联想到手写签名

CA 中心为每个用户签发包含用户公钥的数字证书，而用户私钥只由用户自己保管，CA 中心及每个用户都不知道其他任何用户的私钥。

如果采用各种管理手段和技术手段能够确保不发生以下操作，则在网络上由用户私钥进行的任何加解密操作行为，均可以看作是用户的正常行为：

(1) 用户私钥不会被泄露，任何其他人无法获得该私钥。

(2) 用户私钥不会被劫持，任何其他人不能在违背该用户主观意愿下使用该私钥随意进行加解密操作。

显然，私钥属于用户专有，这种特性很容易让人联想到手写签名。私钥由计算机随机产生，相同的概率很低，具有唯一性；使用私钥对数据进行加解密操作，可看作是“签名动作”（事实上，只有私钥加密属于签名范畴，具体原理可参考后续章节的内容）。使用私钥对电子文档进行加密操作后的结果称作数字签名或电子签名。

尽管电子签名与手写签名具有很高的相似性，但本质上还是存在很多区别的：

(1) 电子签名仅表现为一组代码，需要计算机系统鉴别，无法仅凭视觉来进行辨认。

(2) 电子签名是一种数据，无法以原件形式提交。这对传统的法律证据规则提出了挑战。

(3) 电子签名可通过计算机网络在线签署，可以节省当事人的时间、提高交易效率。

(4) 电子签名需要特殊电子认证方式以确定其真实性。需要经过资质信誉良好的认证机构按照一定的标准，通过计算机系统的核查对电子签名的真实性与有效性予以确认。

(5) 电子签名容易被改动，且修改后不易被发现，可能给电子签名的签署者在电子交易中带来很大的损失。

1.5 电子签名法赋予电子签名与认证法律地位

如果电子文档是一份商务合同，那么用户私钥签署电子签名后是否就成为电子合同呢？如果电子签名没有法律效力，该文档还是一个普通的电子文档，并不是真正意义上的电子合同。可喜的是，期盼已久的《中华人民共和国电子签名法》（以下简称《电子签名法》）终于于 2005 年 4 月 1 日正式施行，该法给电子签名与认证赋予了法律效力，大大增强了电子交易的安全性，保障网上交易者的信心，建立良好的网络信用机制和高效的网上交易途径，对我国电子商务的发展以及网络经济繁荣起到极其重要的促进作用。

《电子签名法》主要规定了电子签名及其相关定义、满足法律法规规定的电子数据、可靠的电子签名和电子认证服务等。需要特别注意的是，并不是所有的电子签名都具有法律效力，只有可靠的电子签名才具有法律效力。

1. 电子签名及相关定义

电子签名，是指数据电文中以电子形式所含、所附用于识别签名人身份并表明签名人认可其中内容的数据。

数据电文，是指以电子、光学、磁或者类似手段生成、发送、接收或者储存的信息。

电子签名人，是指持有电子签名制作数据并以本人身份或者以其所代表的人的名义实施电子签名的人。

电子签名依赖方，是指基于对电子签名认证证书或者电子签名的信赖从事有关活动的人。

电子签名认证证书，是指可证实电子签名人与电子签名制作数据有联系的数据电文或者其他电子记录。

电子签名制作数据，是指在电子签名过程中使用的，将电子签名与电子签名人可靠地联

系起来的字符、编码等数据。

电子签名验证数据，是指用于验证电子签名的数据，包括代码、口令、算法或者公钥等。

2. 满足法律法规规定的数据电文

能够有形地表现所载内容，并可以随时调取查用的数据电文，视为符合法律、法规要求的书面形式。

符合下列条件的数据电文，视为满足法律、法规规定的原件形式要求：

(1) 能够有效地表现所载内容并可供随时调取查用。

(2) 能够可靠地保证自最终形成时起，内容保持完整、未被更改。但是，在数据电文上增加背书以及数据交换、储存和显示过程中发生的形式变化不影响数据电文的完整性。

符合下列条件的数据电文，视为满足法律、法规规定的文件保存要求：

(1) 能够有效地表现所载内容并可供随时调取查用。

(2) 数据电文的格式与其生成、发送或者接收时的格式相同，或者格式不相同但是能够准确表现原来生成、发送或者接收的内容。

(3) 能够识别数据电文的发件人、收件人以及发送、接收的时间。

数据电文有下列情形之一的，视为发件人发送：

(1) 经发件人授权发送的。

(2) 发件人的信息系统自动发送的。

(3) 收件人按照发件人认可的方法对数据电文进行验证后结果相符的。

法律、行政法规规定或者当事人约定数据电文需要确认收讫的，应当确认收讫。发件人收到收件人的收讫确认时，数据电文视为已经收到。

3. 可靠的电子签名

电子签名同时符合下列条件的，视为可靠的电子签名：

(1) 电子签名制作数据用于电子签名时，属于电子签名人专有。

(2) 签署时电子签名制作数据仅由电子签名人控制。

(3) 签署后对电子签名的任何改动能够被发现。

(4) 签署后对数据电文内容和形式的任何改动能够被发现。

可靠的电子签名与手写签名或者盖章具有同等的法律效力。

4. 电子认证服务

电子签名需要第三方认证的，由依法设立电子认证服务提供者提供认证服务。

提供电子认证服务，应当具备下列条件：

(1) 具有与提供电子认证服务相适应的专业技术人员和管理人员。

(2) 具有与提供电子认证服务相适应的资金和经营场所。

(3) 具有符合国家安全标准的技术和设备。

(4) 具有国家密码管理机构同意使用密码的证明文件。

(5) 法律、行政法规规定的其他条件。

从事电子认证服务，应当向国务院信息产业主管部门提出申请，并提交相关材料。国务院信息产业主管部门接到申请后经依法审查，征求国务院商务主管部门等有关部门的意见后，予以许可的，颁发电子认证许可证书；不予许可的，应当书面通知申请人并告知理由。

第2章 PKI 包括哪些内容

2.1 PKI 体系框架

PKI 是 Public Key Infrastructure 的缩写，其主要功能是绑定证书持有者的身份和相关的密钥对（通过为公钥及相关的用户身份信息签发数字证书），为用户提供方便的证书申请、证书作废、证书获取、证书状态查询的途径，并利用数字证书及相关的各种服务（证书发布，黑名单发布，时间戳服务等）实现通信中各实体的身份认证、完整性、抗抵赖性和保密性。

根据数字证书格式及密钥管理方式的不同，PKI 也包括多种模式，如 X.509 模式、PGP 模式、IBE/CPK 模式、EMV 模式等。X.509 模式的 PKI 也称作 PKIX。由于 X.509 标准已经成为数字证书格式的事实标准，因此大部分情况下 PKI 特指 PKIX。如非特殊说明，本文中 PKI 均指 PKIX。

PKI 体系框架主要包括三部分内容。

1. 数字证书与私钥

用户或系统只有拥有自己的公钥和私钥后，才能实现数字签名和加解密功能。由于公钥是随机产生的，因此从公钥无法直接判断属于哪个用户。

为解决公钥与用户映射关系问题，PKI 引入了数字证书，用于建立公钥与用户之间的对应关系。数字证书实际上是一种特殊的文件格式，包含用户身份信息、用户公钥信息和 CA 私钥的数字签名。X.509 标准规定了数字证书的具体格式。数字证书中只包含公钥，并不包含私钥。由于数字证书中包含 CA 私钥的数字签名，因此数字证书具有防伪性。由于数字证书中不包含秘密信息，因此数字证书具有公开性。数字证书作为网络身份证，可有效解决网络世界中“你是谁”的问题。

为保证私钥的安全性，一般将私钥保存在硬件密码设备中（如个人的私钥可保存在 USB Key 或 IC 卡中，系统的私钥可保存在加密机或加密卡中），而且不允许私钥导出硬件密码设备；通过口令、指纹等方式来访问控制该硬件密码设备。如果将私钥保存在硬盘文件中，则需要通过口令进行加密保护，但私钥文件容易被复制。

2. 数字证书的管理

为解决数字证书的签发问题，PKI 引入 CA，对数字证书进行集中签发。CA 是 Certificate Authority 的缩写，字面含义是证书权威，也称作 CA 中心、认证中心。CA 中心拥有自己的公钥和私钥，使用其私钥给用户（包含 CA 中心自己）签发数字证书。

CA 实际是一种特殊的公钥管理中心，为实现数字证书的安全性，对数字证书的全生命周期进行管理，主要包括：数字证书的签发和更新、数字证书的作废（注销、撤销或吊销）、数字证书的冻结（挂失）和解冻、数字证书的查询或下载、数字证书状态查询等。

当私钥丢失时, 如果没有备份和恢复机制, 将导致公钥加密的所有数据都无法解密, 可能给用户带来损失。为解决私钥的备份与恢复问题, PKI 引入了 KMC, 用于对私钥的全生命周期进行管理。KMC 是 Key Management Center 的缩写。用户公私钥对可由 KMC 产生, 提交 CA 签发数字证书后, 将私钥和数字证书同时安全移交给用户, 而 KMC 将私钥留作备份, 可按需要给用户恢复。用户公私钥对也可由用户自己产生(使用软件或密码设备), 在向 CA 中心申请数字证书时, 可将私钥安全提交 KMC 留作备份。

为防止用户身份被冒用, 应保证用户私钥的唯一性, 不允许备份恢复。为防止公钥加密后的数据无法解密, 应提供用户私钥的备份恢复机制。为解决这两种矛盾的应用需求, PKI 引入双证书机制: 签名证书和加密证书。签名证书及私钥只用于签名验签, 不能用于加密解密, 该公私钥对必须由用户自己产生, KMC 不备份签名私钥。加密证书及私钥只用于加密解密, 不能用于签名验签, 该公私钥对必须由 KMC 产生, 且 KMC 对加密私钥进行备份。

CA 中心存储着所有数字证书, 并通过公开服务形式供用户随时查询或下载。为解决数字证书查询或下载的性能问题, 避免 CA 中心成为性能瓶颈, PKI 引入了 LDAP 技术, 通过 LDAP 方式对外提供高速证书查询或下载服务。LDAP 为轻量目录访问协议, 是 Light-weight Directory Access Protocol 的缩写。

当用户私钥泄露后, CA 中心有责任将该用户的证书标记为失效。但用户如何获得对方证书是否失效的状态呢? 为方便用户获得证书状态, PKI 引入了 CRL 和 OCSP 技术。CRL 为证书作废列表, 是 Certificate Revocation List 的缩写, 也称作证书黑名单。OCSP 为在线证书状态协议, 是 Online Certificate Status Protocol 的缩写。

在保证 CA 系统安全性的前提下, 为方便证书业务远程办理、方便证书管理流程与应用系统结合, PKI 引入了 RA, 专门用于对用户提供面对面的证书业务服务, 负责用户证书办理/作废申请、用户身份审核、制作证书并移交用户等功能, 而涉及证书签发时则提交 CA 系统集中处理。RA 是 Registry Authority 的缩写, 又称作 RA 中心、注册中心。有时候, 证书管理流程会与应用业务流程结合, 并不单独体现出来, 如对于网上银行, 证书管理流程就会融合到网银业务流程中。

为保证 CA 系统在数字证书管理方面的规范性、合规性和安全性, PKI 引入电子认证服务机构, 以第三方运营的方式对外提供数字证书相关服务。《电子签名法》规定: “电子签名需要第三方认证的, 由依法设立的电子认证服务提供者提供认证服务。”显然, 电子签名法不仅赋予了电子签名的法律效力, 而且明确了电子认证服务提供者的法律地位。电子认证服务提供者, 又称作电子认证服务机构或 CA 中心, 需要独立运营, 不仅需要满足政策法规的各项要求, 而且需要满足业务运营的业务需求。

3. 数字证书的应用

基于数字证书可实现四种基本安全功能: 身份认证、保密性、完整性和抗抵赖性。

基于证书接口中间件(模块或组件), 应用系统可以很方便地使用数字证书技术, 从而提高应用系统的身份认证强度、保证应用系统中各种敏感数据的保密性、保证应用系统中各种敏感数据和交易记录的完整性、用户各种操作或交易的不可否认性(抗抵赖性)。

PKI 技术经过 30 多年的发展历程, 已经形成比较完善的标准规范体系, 几乎覆盖应用系统的各个方面, 目前很多软件都已经支持数字证书技术, 如操作系统、数据库系统、Web 服

务器、应用服务器等。由于受应用环境多样性和应用技术复杂性的影响或限制，在不同的应用环境和应用技术下，数字证书技术应用的方式可能差异很大。

网上报税和网上银行是数字证书应用的典型案例。国内大部分省份上千万家企业已经通过数字证书在进行网上报税。数字证书（俗称 U 盾）已经成为国内网上银行的标准配置。如果没有数字证书，企业用户将不允许使用网上银行。上亿个人用户已经通过数字证书访问网上银行实现转账或汇款等资金操作。

PKI 体系框架如图 2-1 所示。

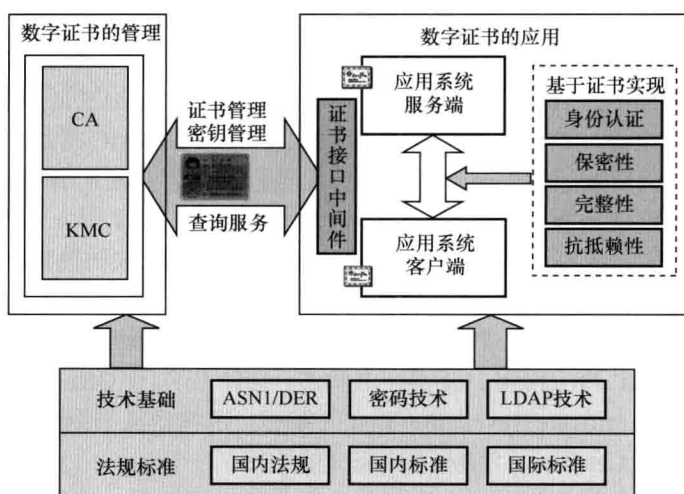


图 2-1 PKI 体系框架

2.2 PKI/数字证书与私钥

1. 数字证书的最初目的是建立公钥与用户之间的对应关系

由于公钥是随机产生的，从公钥无法直接判断属于哪个用户。为解决公钥与用户映射关系问题，PKI 引入数字证书，用于建立公钥与用户之间的对应关系。

数字证书实际是一种特殊的文件格式，包含用户身份信息、用户公钥信息和 CA 私钥的数字签名。用户身份信息由姓名或名称、单位、城市、国家等组成。X.509 标准规定了数字证书的具体格式。

由于数字证书中包含用户身份信息和公钥信息，根据数字证书就可直接判断该公钥属于哪个用户。如某数字证书包含用户信息“诸葛亮”和公钥 PK，则可判断公钥 PK 就是诸葛亮的。

由于数字证书中包含 CA 私钥的数字签名，使用 CA 公钥对数字证书中 CA 私钥的数字签名进行解密处理后，就可立刻判断该数字证书是否被篡改，因此数字证书具有防伪性，于是，数字证书中公钥和用户之间对应关系也值得可信。当然，数字证书防伪性的前提是公钥算法和 CA 私钥都是安全的。

由于数字证书中不包含秘密信息，因此数字证书可公开发布。

数字证书主要内容显示如图 2-2 所示，其中，“使用者”为用户身份信息，“公钥”为用户公钥信息，“有效期”属于附加策略。



图 2-2 数字证书主要内容显示

2. 数字证书可作为网络身份证

现实生活中，公安部为每人颁发一张二代身份证，身份证上包含姓名、性别、出生日期、省份、身份证号、照片等信息。当购票、住店、坐飞机时，身份证持有人（或持证人）只需出示身份证即可证明自己的身份。只有通过以下四个方面的合法性验证，才能完全确认持证人的合法身份：

（1）验证身份证是否伪造。需要专门的二代身份证读写设备来验证。

（2）验证身份证信息是否正确。需要查询身份证数据库，需要公安部门提供；也可由公安部门在发放身份证时保证。

（3）验证身份证是否与持证人一致。需要对比身份证上照片与持证人的长相。

（4）验证身份证是否在黑名单上。需要查询黑名单数据库，如公安部通缉犯清单，需要公安部门提供。

既然数字证书包含用户身份信息，而且具有防伪性且可以公开，那么数字证书持有人（或持证人）在网络世界中能否也像现实生活中一样，只需出示数字证书即可证明自己身份呢？答案是肯定的。如果能通过类似二代身份证的四个方面合法性验证，数字证书就可作为网络身份证。下面来确认一下，数字证书能否通过四个方面的合法性验证：

（1）验证数字证书是否伪造。可通过 CA 公钥验证。

（2）验证数字证书中信息是否正确。可由 CA 中心在签发数字证书时保证。

（3）验证数字证书是否与持证人一致。可要求持证人使用私钥对特定数据进行加密或签名，然后使用数字证书中的公钥来解密该数据或验签，从而可验证持证人是否持有与数字证书中的公钥对应的私钥。

(4) 验证数字证书是否在黑名单上。可通过查询黑名单数据库来实现，但需要 CA 中心提供黑名单。

因此，数字证书作为网络身份证，可有效解决网络世界中“你是谁”的问题。由于数字证书采用公钥密码技术实现，从技术上保证了每个人“钥匙”的唯一性，既不重复也无法复制，所以数字证书比现实生活中的各种钥匙安全很多。

3. 数字证书可附加很多策略

由于数字证书具有防伪性和公开性，因此数字证书中不仅可包含用户身份信息和用户公钥信息，而且还可以附加其他策略信息，这些信息同样具有防伪性和公开性。

X.509 数字证书标准中规定常用策略信息主要包括：

- (1) 有效期。包括起始时间和终止时间。
- (2) 密钥用途。表示该公钥和私钥用于什么目的，如数据签名、数据加密、签发证书、用于安全电子邮件等。
- (3) 别名。包括持证人别名和 CA 中心别名。
- (4) 黑名单地址。
- (5) 是否 CA 中心的数字证书。如果是 CA 中心的数字证书，则规定能签发几级证书。

4. CA 中心的数字证书

用户在验证数字证书是否被篡改时，必须首先获得 CA 中心的公钥。为方便用户识别 CA 中心的公钥，CA 中心也为自己签发数字证书。该证书包含 CA 中心公钥、CA 中心身份信息和 CA 中心私钥的签名。CA 中心的数字证书有时也称作 CA 证书。

由于 CA 证书是验证其他证书合法性的前提和基础，所以 CA 证书的正确性至关重要。尽管可通过网络直接下载获取 CA 证书，但为避免被欺骗，需要通过其他方式验证 CA 证书的正确性。可通过电话方式、官方报纸或官方电视公布 CA 证书的摘要值，用户通过核对该摘要值即可确认该证书的正确性。

5. 私钥

对于公钥密码技术，公钥和私钥是成对出现的。公钥以数字证书形式存在，可以公开，但私钥必须保密。为保证私钥的安全性，一般将私钥保存在硬件密码设备中（如个人的私钥可保存在 USB Key 或 IC 卡中，系统的私钥可保存在加密机或加密卡中），而且不允许私钥导出硬件密码设备；通过口令、指纹等方式来访问控制该硬件密码设备。如果将私钥保存在硬盘文件中，则需要通过口令进行加密保护，但私钥文件容易被复制。

为保证私钥的唯一性，可只允许在硬件密码设备内产生公私钥对，通过硬件技术保证私钥无法导出，然后只导出公钥提交 CA 中心签发数字证书。

2.3 PKI/CA 与 KMC

1. CA 最初目的是签发数字证书

为解决数字证书的签发问题，PKI 引入 CA，用于对数字证书进行集中签发。CA 是 Certificate Authority 的缩写，字面含义是证书权威，也称作 CA 中心、认证中心。CA 中心拥

有自己的公钥和私钥，使用其私钥给用户（包含 CA 中心自己）签发数字证书，具体签发过程如下：

- （1）将用户身份信息和用户公钥信息，按照特定格式组成数据 D。
- （2）选择摘要算法对数据 D 进行计算得到摘要 H。
- （3）使用 CA 私钥对摘要 H 进行加密得到数字签名 S。
- （4）将用户身份信息、用户公钥信息和数字签名 S，按照特定格式就可组成数字证书。

2. CA 应对数字证书的全生命周期进行管理

CA 实际是一种特殊的公钥管理中心，为实现数字证书的安全性，应对数字证书的全生命周期进行管理，主要包括：

- （1）数字证书的签发和更新。证书签发主要针对新用户；证书更新主要针对已有证书用户，更新时公钥可以改变，也可以不变。
- （2）数字证书的作废（注销、撤销或吊销）。证书作废后将成无效证书，永远不能使用。
- （3）数字证书的冻结（挂失）和解冻。证书冻结后将成无效证书，但可以通过解冻恢复成有效状态。
- （4）数字证书的查询或下载。应提供公开服务方式，允许用户根据条件随时查询证书并下载。常用服务方式有 HTTP 和 LDAP。
- （5）数字证书状态查询。应提供公开服务方式，允许用户随时查询证书状态，以便判断该证书是否处于有效状态。常用服务方式有 CRL 和 OCSP。

3. KMC 用于管理私钥

如果使用公钥（数字证书）对数据进行加密，则只有与公钥对应的私钥才能进行解密，从而可实现用户数据的保密性。当私钥丢失时，如果没有备份和恢复机制，将导致公钥加密的所有数据都无法解密，可能给用户带来损失。

为解决私钥的备份与恢复问题，PKI 引入了 KMC，用于对私钥的全生命周期进行管理。KMC 是 Key Management Center 的缩写。

用户公私钥对可由 KMC 产生，提交 CA 签发数字证书后，将私钥和数字证书同时安全移交给用户，而 KMC 将私钥留作备份，可按需要给用户恢复。用户公私钥对也可由用户自己产生（使用软件或密码设备），在向 CA 中心申请数字证书时，可将私钥安全提交 KMC 留作备份。

4. 双证书机制

为防止用户身份被冒用，应保证用户私钥的唯一性，不允许备份恢复。为防止公钥加密后的数据无法解密，应提供用户私钥的备份恢复机制。

为解决这两种矛盾的应用需求，PKI 引入双证书机制：签名证书和加密证书。

签名证书及私钥只用于签名验签，不能用于加密解密；为保证签名私钥的唯一性，该公私钥对必须由用户自己产生，同时采用硬件技术（如只允许在硬件密码设备内产生公私钥对、私钥不允许导出等）保证无法复制，在证书签发过程中 CA 中心并不知道该私钥，只对其公钥进行操作。该私钥没有备份，因此永远不能恢复。

加密证书及私钥只用于加密解密，不能用于签名验签。为实现解密私钥的备份恢复机制，

该公私钥对必须由 KMC 产生，由 CA 签发完数字证书后安全移交给用户，用户应采用硬件技术保证该私钥的安全性。KMC 可根据需要恢复该私钥。

5. LDAP

CA 中心存储着所有数字证书，并通过公开服务形式供用户随时查询或下载。为解决数字证书查询或下载的性能问题，避免 CA 中心成为性能瓶颈，PKI 引入了 LDAP 技术，通过 LDAP 方式对外提供高速证书查询或下载服务。

专业的关系型数据库管理系统（如 Oracle、DB2 等）专门对结构化数据进行管理，应用系统只需通过 SQL 语句（报文协议）发送各种数据管理指令，而具体管理工作完全由数据库管理系统负责。尽管这种数据管理方式大大简化了应用系统的复杂度，但由于其读写功能相对均衡，很难满足高性能的查询服务。为获得高性能的查询服务，需要对数据管理的读取功能进行特殊优化，于是出现了目录服务技术（DAP，Directory Access Protocol）。目录服务技术对查询功能进行优化，比修改操作快 10 倍以上，适合快速响应和大容量查询服务。DAP 技术通过目录树方式对数据进行管理，应用系统只需通过 X.500 协议就能对数据进行快速查询。

由于 X.500 协议过于复杂，实现成本很高，于是国际组织对 X.500 进行了简化，形成 LDAP 标准，而且 LDAP 支持 TCP/IP 协议，更适合于互联网领域使用。LDAP 为轻量目录访问协议，是 Light-weight Directory Access Protocol 的缩写。

由于数字证书是可以公开的，CA 中心也可以将其签发的数字证书存储到其他地方，供用户查询或下载。

6. CRL 和 OCSP

当用户私钥泄露后，CA 中心有责任将该用户的证书标记为失效。但用户如何获得对方证书是否失效的状态呢？为方便用户获得证书状态，PKI 引入了 CRL 和 OCSP 技术。

（1）CRL（Certificate Revocation List）

CRL 为证书作废列表，是 Certificate Revocation List 的缩写，也称作证书黑名单，是一种特殊的文件格式，包含所有失效的证书清单、下次 CRL 生成时间和 CA 中心私钥的签名。

CA 中心定期生成 CRL，并将下次生成时间写入 CRL 中，方便用户按时定期下载 CRL。跟数字证书类似，CRL 也是通过 CA 中心私钥的签名来保证 CRL 无法被篡改的。用户只需定期获取 CRL 后，就可在本地脱机验证证书是否失效，在下次 CRL 生成时间之前无需联系 CA 中心。

由于 CRL 是可以公开的，CA 中心也可以将其签发的 CRL 存储到其他地方，供用户查询或下载。CRL 也可以发布到 LDAP 中，提供高速下载服务。

（2）OCSP（Online Certificate Status Protocol）

当 CA 中心将私钥已泄露的用户证书标记为失效后，如果还未到下次 CRL 生成时间，此时其他用户通过最新 CRL 并不知道该用户证书已经失效，依然将该用户当作有效用户继续进行各种保密通信和合法交易，因此可能会造成一定的损失。

为解决 CRL 滞后的缺陷，避免给高实时性或高风险交易造成重大损失，PKI 引入了 OCSP，对用户提供实时的证书状态查询服务。

OCSP 为在线证书状态协议，是 Online Certificate Status Protocol 的缩写。当用户需要实时查询对方证书是否有效时，可通过 OCSP 协议实时访问 CA 中心获得对方证书的当前状态。

7. RA

在保证 CA 系统安全性的前提下, 为方便证书业务远程办理、方便证书管理流程与应用系统结合, PKI 引入了 RA, 专门用于对用户面对面的证书业务服务, 负责用户证书办理/作废申请、用户身份审核、制作证书并移交用户等功能, 而涉及证书签发时则提交 CA 系统集中处理。RA 是 Registry Authority 的缩写, 又称作 RA 中心、注册中心。有时候, 证书管理流程会与应用业务流程结合, 并不单独体现出来, 如对于网上银行, 证书管理流程就会融合到网银业务流程中。

RA 可以部署多个, 既可以单独部署, 也可以与应用系统集成。

CA 主要模块组成如图 2-3 所示。

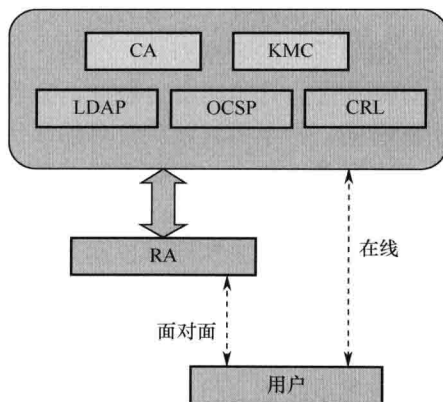


图 2-3 CA 主要模块组成

2.4 PKI/应用

1. 基于数字证书可实现四种基本安全功能

(1) 身份认证。

网上交易的双方很可能素昧平生, 相隔千里。要使交易成功进行, 首先要能确认对方的身份, 对商家要考虑客户不能是骗子, 而客户也会担心网上的商店是不是一个玩弄欺诈的黑店, 因此能方便而可靠地确认对方身份是交易成功的前提。对于为顾客或用户开展服务的银行、信用卡公司和销售商店, 为了做到安全、保密、可靠地开展服务活动, 都要进行身份认证的工作。对销售商店来说, 他们对顾客刷卡消费所用的信用卡的真伪是不知道的, 商店只能把信用卡的身份确认工作完全交给银行来完成。

数字证书可作为网络身份证, 在网络世界中进行交互时, 证书持有人(持证人)只需出示数字证书, 对方通过判断该证书是否伪造、持证人是否拥有对应的私钥、该证书是否被作废或冻结等内容即可验证该持证人的身份是否合法, 从而很方便地实现高强度的身份认证功能。

(2) 保密性。

交易中的商务信息均有保密的要求。如信用卡的账号和用户名被人知悉, 就可能被盗用, 订货和付款的信息被竞争对手获悉, 就可能丧失商机。

使用数字证书中的公钥对静态数据(如文档)或动态数据(如交易报文)进行加密, 只有持证人才能使用对应的私钥进行解密, 从而实现各种敏感数据的保密性。

(3) 完整性。

为保障交易的严肃和公正, 交易的文件不应被修改, 如合同或订单。卖方收到订单后, 如发现商品大幅涨价, 若能私自改动合同, 将商品价格或订购数量修改, 就可大幅受益, 那么买方可能就会蒙受损失。

使用私钥对静态数据(如文档)或动态数据(如交易报文)进行签名, 使用对应的数字证书中的公钥就能验证该数据是否被篡改, 从而实现各种敏感数据和交易记录的完整性。

(4) 抗抵赖性（不可否认性）。

由于商情的千变万化，交易一旦达成是不能被否认的，否则必然会损害一方的利益。例如订购黄金，订货时金价较低，但收到订单后，金价上涨了，如果销售方能否认收到订单的实际时间，甚至否认收到订单的事实，则订货方就会蒙受损失。

在交易过程中，可要求对方使用其私钥对交易数据进行签名，交易完成后可将交易数据及对方签名存储起来，一旦发生交易纠纷，就可调出所存储的交易数据和对方签名，使用对方数字证书中的公钥即可证明该交易是对方进行的，从而实现交易过程的抗抵赖性。

2. 数字证书如何与应用系统结合

基于证书接口中间件（模块或组件），应用系统可以很方便地使用数字证书技术，从而提高应用系统的身份认证强度、保证应用系统中各种敏感数据的保密性、保证应用系统中各种敏感数据和交易记录的完整性、用户各种操作或交易的不可否认性。

PKI 技术经过 30 多年的发展历程，已经形成比较完善的标准规范体系，几乎覆盖应用系统的各个方面，目前很多软件都已经支持数字证书技术，如操作系统、数据库系统、Web 服务器、应用服务器等。由于受应用环境多样性和应用技术复杂性的影响或限制，在不同的应用环境和应用技术下，数字证书技术应用的方式可能差异很大。

当前主流的数字证书应用技术主要包括：Windows 域登录、操作系统登录、电子文件加密、安全电子邮件、电子印章、代码签名、时间戳、WTLS 应用、SSL/TLS 应用（如 Web 网站安全、SSL VPN）、IP Sec 应用（如 IPSec VPN）等。

3. 数字证书典型应用

(1) 网上报税。

随着 IT 技术的迅速发展及其在税收领域的广泛应用，网上税收已成为不可逆转的发展趋势，其主要存在两类安全隐患或问题：一是网络安全问题。重点是身份认证，目前采用的用户名/口令机制容易被他人假冒身份；二是纸质报表的事后报表问题。只有加盖企业公章的纸质报表才能作为法律凭证。

电子签名法赋予数字签名法律效力后，数字证书技术就可有效解决上述两个问题，使得企业网上报税成为现实，同时提高了企业和税务部门的工作效率。

事实上，国内大部分省份上千万家企业已经通过数字证书进行网上报税。

(2) 网上银行。

网上银行是商业银行基于互联网为客户提供安全、实时的银行业务服务。作为一种全新的银行客户服务渠道，客户可以不必亲自去银行办理业务，只要能够上网，无论在家里、办公室，还是在旅途中，都能够每天 24 小时安全便捷地管理自己的资产，或者办理查询、转账、缴费等银行业务。

网上银行以 Internet 等开放式网络环境传输交易数据，而且涉及用户资金转移等敏感信息，所以在用户的身份认证、资金的秘密传输以及数据的完整性方面，存在许多安全问题。网上银行服务提供者首先需要确定自己的系统不会受到网络黑客的入侵，造成秘密信息泄露、业务损失或服务中断。对用户而言，必须确认在网络上输入的秘密信息不会被盗用、输入的交易资料不会被篡改并且能正确迅速地传送到接收端系统。

基于数字证书技术，网上银行能有效解决用户身份认证、敏感数据保密性、交易数据完

整性和交易操作不可抵赖性问题，极大地方便了银行企业客户和个人客户。

数字证书（俗称 U 盾）已经成为国内网上银行的标准配置。如果没有数字证书，企业用户将不允许使用网上银行。上亿个人用户已经通过数字证书访问网上银行实现转账或汇款等资金操作。

2.5 PKI/运营

《电子签名法》规定：“电子签名需要第三方认证的，由依法设立电子认证服务提供者提供认证服务。”显然，电子签名法不仅赋予了电子签名的法律效力，而且明确了电子认证服务提供者的法律地位。电子认证服务提供者又称作电子认证服务机构或 CA 中心。

1. 电子认证服务机构需满足政策法规的各项要求

《电子签名法》（2004 年版）规定：提供电子认证服务，应当具备下列条件：

- (1) 具有与提供电子认证服务相适应的专业技术人员和管理人员。
- (2) 具有与提供电子认证服务相适应的资金和经营场所。
- (3) 具有符合国家安全标准的技术和设备。
- (4) 具有国家密码管理机构同意使用密码的证明文件。
- (5) 法律、行政法规规定的其他条件。

《电子认证服务管理办法》（2009 年版）规定：电子认证服务机构应当具备下列条件：

- (1) 具有独立的企业法人资格。
- (2) 具有与提供电子认证服务相适应的人员。从事电子认证服务的专业技术人员、运营管理人员、安全管理人员和客户服务人员不少于三十名，并且应当符合相应岗位技能要求。
- (3) 注册资本不低于人民币三千万元。
- (4) 具有固定的经营场所和满足电子认证服务要求的物理环境。
- (5) 具有符合国家有关安全标准的技术和设备。
- (6) 具有国家密码管理机构同意使用密码的证明文件。
- (7) 法律、行政法规规定的其他条件。

《电子政务电子认证服务管理办法》（2009 年版）规定：电子认证服务机构应当具备以下条件：

- (1) 事业法人或者取得电子认证服务许可的国有控股企业法人。
- (2) 具有经国家密码管理局批准的电子政务电子认证基础设施。
- (3) 具有与从事认证服务相适应的专业技术人员、运行维护人员、安全管理人员和服务人员。
- (4) 具有与认证服务相适应的运行服务、应用支持和安全保障等机制。
- (5) 法律法规规定的其他条件。

《卫生系统电子认证服务管理办法》（2009 年版）规定：电子认证服务机构面向卫生系统提供电子认证服务应当具备以下必要条件：

- (1) 取得工业和信息化部颁发的《电子认证服务许可证》。
- (2) 符合《电子政务电子认证体系建设总体规划》（国密局联字〔2007〕2 号）中关于电

子认证体系建设的相关要求。

(3) 具有符合《卫生系统电子认证服务规范》、《卫生系统数字证书格式规范》、《卫生系统数字证书介质技术规范》、《卫生系统数字证书应用集成规范》和《卫生系统数字证书服务管理平台接入规范》等的电子认证服务体系。

(4) 符合法律、行政法规规定的其他条件。

2. 电子认证服务机构需满足业务运营的业务需求

(1) 认证业务方面

应包括两类业务服务：用户证书服务和用户证书密钥服务。用户证书服务主要包括：证书申请与审核、证书签发、证书存储与发布、证书更新、证书撤销、证书冻结、证书状态查询、证书归档等。用户证书密钥服务主要包括：用户证书密钥的产生/传递/存储、用户证书私钥激活数据的产生/传递/存储、用户证书私钥恢复、用户证书密钥对的更新、用户证书私钥的归档和销毁等。

为保证认证业务的安全性，应加强业务流程的管控，严格制定各种业务流程或策略，主要包括证书申请审核流程和规范、证书更新的自动审核策略、证书撤销管理策略和流程、证书冻结策略和流程、证书状态查询服务策略、证书归档策略、用户证书密钥和私钥激活数据的传递策略、用户证书私钥恢复的管理规定和流程、用户证书私钥归档/销毁流程等。

(2) 认证系统方面

认证系统功能方面，应符合 GB/T 25056-2010（信息安全技术 证书认证系统密码及其相关安全技术规范）中的要求，能够提供用户证书的申请、签发、存储、发布、更新、撤销、冻结、状态查询、归档以及用户证书密钥管理等功能。

认证系统运行方面，应保证各层面的技术和管理安全性，主要包括：网络系统安全、主机系统安全、系统冗余与备份、系统运营维护安全管理、密码设备安全管理、CA 密钥和证书管理等。其中，系统冗余包括：网络链路冗余、主机冗余、电源冗余与后备发电等。系统备份包括：软件与数据备份、硬件设备备份等。系统运营维护安全管理包括：系统权限管理、系统操作管理、系统变更和升级、账户与口令管理、系统安全监控等。密码设备安全管理包括：认证系统密码设备管理、用户密码设备管理等。CA 密钥和证书管理包括：CA 密钥的生成和存储、CA 私钥激活数据的管理、CA 密钥的使用、CA 密钥的备份与恢复、CA 密钥的销毁、CA 证书的创建和发布、CA 证书的更新、CA 证书的撤销、CA 密钥和证书的归档等。

(3) 物理环境与设施方面。

主要包括：运营场地、运营区域划分及要求、安全监控系统、环境保护与控制设施、支撑设备、场地访问安全管理、场地监控安全管理、注册机构场地安全等。

其中，运营区域分为：公共区（控制区）、服务区（限制区）、管理区（敏感区）、核心区（机密区）等。安全监控系统包括：门禁、入侵检测、监控录像等。环境保护与控制设施包括：空气和温湿度控制、防雷击和接地、静电防护、水患防治、消防设施等。支撑设备包括：供配电系统、照明等。

(4) 组织与人员管理方面。

主要包括：职能与角色设置、安全组织、人员安全管理等。

其中，职能与角色设置中，必要的职能部门包括：安全策略管理部门、安全管理部门、

运营管理部门、人事管理部门、认证服务部门、技术服务部门等；必要的岗位角色包括：安全策略管理组织负责人、安全管理人员、密钥管理员、系统维护员、鉴别与验证员、客户档案管理员、客户服务员、审计员、人事经理等。人员安全管理包括：人员安全策略、可信背景调查、人员可信保障、人员异动处理等。

（5）文档、记录与介质管理方面。

主要包括：文档管理、记录管理、介质管理等。

其中，文档管理包括：文档归类、注册机构的文档、人员与制度、文档保存、使用控制、文档销毁等。介质管理包括：保管、使用、销毁等。

文档可分为以下几类：企业管理类、安全策略类、运营管理类、客户类。记录可分为以下几类：物理场地与设施日常管理记录、安全监控记录、密码设备管理记录、密码设备操作使用记录、CA 证书和密钥维护记录、认证系统运行日志、运营网络安全监测记录、认证系统操作日志、认证服务记录等。

（6）业务连续性方面。

主要包括：业务连续性计划、应急处理预案、灾难恢复计划、灾备中心等。

（7）审计与改进方面。

审计包括：系统审计、运营审计、注册机构审计等。系统审计的目的是发现电子认证服务机构业务系统运行和操作中存在的风险和问题，并依此采取相应的措施和手段，防止安全事故的发生，杜绝问题再次发生。运营审计是为了检查、确认电子认证服务机构是否按照其电子认证业务规则、业务规范、管理制度和安全策略开展业务，发现存在的问题。注册机构审计内容主要包括法律法规符合性、安全运营管理、风险管理等，检查其是否严格按照相关的安全策略及运营管理规范开展业务活动。

在审计时发现与安全要求不相符的事项，应按照相应的调整策略和规程进行适当调整，必要时对调整后的事项进行评估，然后实施调整后的事项。

2.6 PKI/法规与标准

PKI 体系涉及密码技术、电子签名、电子认证、IC 卡技术、Java 技术等，须遵循相应的法规与标准。

2.6.1 国内法规

1. 商用密码相关的政策法规

- （1）商用密码管理条例，1999 年
- （2）商用密码科研管理规定，2005 年
- （3）商用密码产品生产管理规定，2005 年
- （4）商用密码产品销售管理规定，2005 年
- （5）商用密码产品使用管理规定，2007 年
- （6）境外组织和个人在华使用密码产品管理办法，2007 年
- （7）密码产品和含有密码技术的设备进口管理目录，2009 年和 2014 年

2. 电子签名与认证服务相关的政策法规

- (1) 电子签名法, 2004 年
- (2) 电子认证服务管理办法, 2005 年和 2009 年
- (3) 电子认证服务密码管理办法, 2005 年和 2009 年
- (4) 电子政务电子认证服务管理办法, 2009 年
- (5) 卫生系统电子认证服务管理办法, 2009 年
- (6) 电子招标投标办法, 2013 年
- (7) 非金融机构支付服务业务系统检测认证管理规定, 2011 年
- (8) 电子银行业务管理办法, 2006 年
- (9) 关于落实“两个减负”优化纳税服务工作的意见, 2007 年

2.6.2 国内标准

1. 通用性标准

- (1) GM/T 0001-2012 《祖冲之序列密码算法》
- (2) GM/T 0002-2012 《SM4 分组密码算法》(原 SMS4 分组密码算法)
- (3) GM/T 0003-2012 《SM2 椭圆曲线公钥密码算法》
- (4) GM/T 0004-2012 《SM3 密码杂凑算法》
- (5) GM/T 0005-2012 《随机性检测规范》
- (6) GM/T 0006-2012 《密码应用标识规范》
- (7) GM/T 0008-2012 《安全芯片密码检测准则》
- (8) GM/T 0009-2012 《SM2 密码算法使用规范》
- (9) GM/T 0010-2012 《SM2 密码算法加密签名消息语法规则》
- (10) GM/T 0011-2012 《可信计算 可信密码支撑平台功能与接口规范》
- (11) GM/T 0012-2012 《可信计算 可信密码模块接口规范》
- (12) GM/T 0013-2012 《可信计算 可信密码模块符合性检测规范》
- (13) GM/T 0014-2012 《数字证书认证系统密码协议规范》
- (14) GM/T 0015-2012 《基于 SM2 密码算法的数字证书格式规范》
- (15) GM/T 0016-2012 《智能密码钥匙密码应用接口规范》
- (16) GM/T 0017-2012 《智能密码钥匙密码应用接口数据格式规范》
- (17) GM/T 0018-2012 《密码设备应用接口规范》
- (18) GM/T 0019-2012 《通用密码服务接口规范》
- (19) GM/T 0020-2012 《证书应用综合服务接口规范》
- (20) GM/T 0021-2012 《动态口令密码应用技术规范》
- (21) GM/T 0022-2014 《IPSec VPN 技术规范》
- (22) GM/T 0023-2014 《IPSec VPN 网关产品规范》
- (23) GM/T 0024-2014 《SSL VPN 技术规范》
- (24) GM/T 0025-2014 《SSL VPN 网关产品规范》
- (25) GM/T 0026-2014 《安全认证网关产品规范》
- (26) GM/T 0027-2014 《智能密码钥匙技术规范》

- (27) GM/T 0028-2014《密码模块安全技术要求》
- (28) GM/T 0029-2014《签名验签服务器技术规范》
- (29) GM/T 0030-2014《服务器密码机技术规范》
- (30) GM/T 0031-2014《安全电子签章密码技术规范》
- (31) GM/T 0032-2014《基于角色的授权管理与访问控制技术规范》
- (32) GM/T 0033-2014《时间戳接口规范》
- (33) GM/T 0034-2014《基于 SM2 密码算法的证书认证系统密码及其相关安全技术规范》
- (34) GM/T 0035-2014《射频识别系统密码应用技术要求》
- (35) GM/T 0036-2014《采用非接触卡的门禁系统密码应用技术指南》
- (36) GM/T 0037-2014《证书认证系统检测规范》
- (37) GM/T 0038-2014《证书认证密钥管理系统检测规范》
- (38) GM/Z 0001-2013《密码术语》
- (39) GB/T 25056-2010《信息安全技术 证书认证系统密码及其相关安全技术规范》
- (40) GB/T 28447-2012《信息安全技术 电子认证服务机构运营管理规范》

2. 行业性标准

- (1)《卫生系统电子认证服务规范（试行）》，2010 年
- (2)《卫生系统数字证书应用集成规范（试行）》，2010 年
- (3)《卫生系统数字证书格式规范（试行）》，2010 年
- (4)《卫生系统数字证书介质技术规范（试行）》，2010 年
- (5)《卫生系统数字证书服务管理平台接入规范（试行）》，2010 年
- (6) JR/T 0068-2012《网上银行系统信息安全通用规范》
- (7)《电子招标投标系统技术规范 第1部分 交易平台技术规范》，2013 年
- (8)《中国人民银行信息系统电子认证应用指引》，2011 年
- (9)《非金融机构支付业务设施技术要求 V3》，2013 年

2.6.3 国际标准

1. PKCS 系列标准

PKCS 是公钥密码标准（Public Key Cryptography Standards）的缩写，它是由美国 RSA 实验室与遍布全球的安全系统开发者一起合作制定的一组规范，以推动公钥密码的发展。最早发布的 PKCS 文档是早期一群公钥技术使用者在 1991 年召开的一次会议上的成果，目前 PKCS 规范已被广泛引用和实施，部分 PKCS 规范已经成为多个国际组织正式或事实上的标准，如 ANSI X9 文档系列、PKIX、SET、S/MIME、SSL 等。PKCS 系列主要包括以下标准：

PKCS #1: RSA Cryptography Standard（RSA 密码标准）。

PKCS #2: 已并入 PKCS #1，不存在。

PKCS #3: Diffie-Hellman Key Agreement Standard（DH 密钥协商标准）。

PKCS #4: 已并入 PKCS #1，不存在。

PKCS #5: Password-Based Cryptography Standard（基于口令的密码标准）。

- PKCS #6: Extended-Certificate Syntax Standard (扩展的证书语法标准)。
- PKCS #7: Cryptographic Message Syntax Standard (密码消息语法标准)。
- PKCS #8: Private-Key Information Syntax Standard (私钥信息语法标准)。
- PKCS #9: Selected Attribute Types (可供选择的属性类型)。
- PKCS #10: Certification Request Syntax Standard (证书请求语法标准)。
- PKCS #11: Cryptographic Token Interface Standard (密码 Token 接口标准)。
- PKCS #12: Personal Information Exchange Syntax Standard (个人信息交换语法标准)。
- PKCS #13: Elliptic Curve Cryptography Standard (椭圆曲线密码标准), 正在制定中。
- PKCS #14: Pseudo-random Number Generation (伪随机数生成算法 PRNG), 正在制定中。
- PKCS #15: Cryptographic Token Information Format Standard (密码 Token 信息格式标准)。

2. ISO/IEC 7816 系列标准

ISO/IEC 7816 系列标准规定了 IC 卡 (Integrated Circuit Cards) 相关技术标准, 由 ISO (International Organization for Standardization) 和 IEC (International Electrotechnical Commission) 组织共同维护, 目前包括 14 个部分。

ISO 7816-1: Physical characteristics (卡的物理特性)。

ISO 7816-2: Cards with contacts: Dimensions and location of the contacts (触点集成电路卡: 触点的尺寸与位置)。

ISO 7816-3: Cards with contacts: Electrical interface and transmission protocols (触点集成电路卡: 电信号和传输协议)。

ISO 7816-4: Organization, security and commands for interchange (用于交换的结构、安全和命令)。

ISO 7816-5: Registration of application providers (卡应用提供者注册)。

ISO 7816-6: Interindustry data elements for interchange (行业间数据元)。

ISO 7816-7: Interindustry commands for Structured Card Query Language (SCQL) (用于结构化卡查询语言 (SCQL) 的行业间命令)。

ISO 7816-8: Commands for security operations (与安全相关的行业间命令)。

ISO 7816-9: Commands for card management (用于卡管理的命令)。

ISO 7816-10: Electronic signals and answer to reset for synchronous cards (同步卡的电信号和复位应答)。

ISO 7816-11: Personal verification through biometric methods (通过生物识别方法的个人验证)。

ISO 7816-12: Cards with contacts: USB electrical interface and operating procedures (带触点集成电路卡: USB 电气接口及操作规程)。

ISO 7816-13: Commands for application management in multi-application environment (在多应用环境中用于应用管理的命令)。

ISO 7816-15: Cryptographic information application (密码信息应用)。

3. IETF PKIX 系列标准

在 IETF (Internet Engineering Task Force) 内有 PKIX (Public-Key Infrastructure (X.509))

工作组,负责与 X.509 有关的规范管理。PKIX 工作组从 1995 年 10 月 26 日开始启动,到 2013 年 10 月 31 日关闭。在近 20 年间,发布的 RFC 规范主要包括:

RFC 2459: Internet X.509 Public Key Infrastructure Certificate and CRL Profile, 1999-01, 被 RFC 3280 替代。

RFC 2510: Internet X.509 Public Key Infrastructure Certificate Management Protocols, 1999-03, 被 RFC 4210 替代。

RFC 2511: Internet X.509 Certificate Request Message Format, 1999-03, 被 RFC 4211 替代。

RFC 2527: Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework, 1999-03, 被 RFC 3647 替代。

RFC 2528: Internet X.509 Public Key Infrastructure Representation of Key Exchange Algorithm (KEA) Keys in Internet X.509 Public Key Infrastructure Certificates, 1999-03。

RFC 2559: Internet X.509 Public Key Infrastructure Operational Protocols—LDAPv2, 1999-04, 被 RFC 3494 替代。

RFC 2560: X.509 Internet Public Key Infrastructure Online Certificate Status Protocol—OCSP, 1999-06, 被 RFC 6960 替代, 由 RFC 6277 更新。

RFC 2585: Internet X.509 Public Key Infrastructure Operational Protocols: FTP and HTTP, 1999-05。

RFC 2587: Internet X.509 Public Key Infrastructure LDAPv2 Schema, 1999-06, 被 RFC 4523 替代。

RFC 2797: Certificate Management Messages over CMS, 2000-04, 被 RFC 5272 替代。

RFC 2875: Diffie-Hellman Proof-of-Possession Algorithms, 2000-07, 被 RFC 6955 替代。

RFC 3029: Internet X.509 Public Key Infrastructure Data Validation and Certification Server Protocols, 2001-02。

RFC 3039: Internet X.509 Public Key Infrastructure Qualified Certificates Profile, 2001-01, 被 RFC 3739 替代。

RFC 3161: Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP), 2001-08, 被 RFC 5816 更新。

RFC 3279: Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, 2002-05, 被 RFC 4055、RFC 4491、RFC 5480、RFC 5758 更新。

RFC 3280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, 2002-05, 被 RFC 5280 替代, 由 RFC 4325、RFC 4630 更新。

RFC 3281: An Internet Attribute Certificate Profile for Authorization, 2002-05, 被 RFC 5755 替代。

RFC 3379: Delegated Path Validation and Delegated Path Discovery Protocol Requirements, 2002-09。

RFC 3628: Policy Requirements for Time-Stamping Authorities (TSAs), 2003-11。

RFC 3647: Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework, 2003-11。

RFC 3709: Internet X.509 Public Key Infrastructure: Logotypes in X.509 Certificates, 2004-02, 被 RFC 6170 更新。

RFC 3739: Internet X.509 Public Key Infrastructure: Qualified Certificates Profile, 2004-03。

RFC 3770: Certificate Extensions and Attributes Supporting Authentication in Point-to-Point Protocol (PPP) and Wireless Local Area Networks (WLAN), 2004-05, 被 RFC 4334 替代。

RFC 3779: X.509 Extensions for IP Addresses and AS Identifiers, 2004-06。

RFC 3820: Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile, 2004-06。

RFC 4043: Internet X.509 Public Key Infrastructure Permanent Identifier, 2005-05。

RFC 4055: Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, 2005-06, 被 RFC 5756 更新。

RFC 4059: Internet X.509 Public Key Infrastructure Warranty Certificate Extension, 2005-05。

RFC 4158: Internet X.509 Public Key Infrastructure: Certification Path Building, 2005-09。

RFC 4210: Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP), 2005-09, 被 RFC 6712 更新。

RFC 4211: Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF), 2005-09。

RFC 4212: Alternative Certificate Formats for the Public-Key Infrastructure Using X.509 (PKIX) Certificate Management Protocols. October 2005。

RFC 4262: X.509 Certificate Extension for Secure/Multipurpose Internet Mail Extensions (S/MIME) Capabilities. December 2005。

RFC 4325: Internet X.509 Public Key Infrastructure Authority Information Access Certificate Revocation List (CRL) Extension, 2005-12, 被 RFC 5280 替代。

RFC 4334: Certificate Extensions and Attributes Supporting Authentication in Point-to-Point Protocol (PPP) and Wireless Local Area Networks (WLAN), 2006-02。

RFC 4386: Internet X.509 Public Key Infrastructure Repository Locator Service, 2006-02。

RFC 4387: Internet X.509 Public Key Infrastructure Operational Protocols: Certificate Store Access via HTTP, 2006-02。

RFC 4476: Attribute Certificate (AC) Policies Extension, 2006-05。

RFC 4491: Using the GOST R 34.10-94, GOST R 34.10-2001, and GOST R 34.11-94 Algorithms with the Internet X.509 Public Key Infrastructure Certificate and CRL Profile, 2006-05。

RFC 4510: Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map. June 2006。

RFC 4511: Lightweight Directory Access Protocol (LDAP): The Protocol. June 2006。

RFC 4512: Lightweight Directory Access Protocol (LDAP): Directory Information Models, June 2006。

RFC 4513: Lightweight Directory Access Protocol (LDAP): Authentication Methods and Security Mechanisms, June 2006。

RFC 4514: Lightweight Directory Access Protocol (LDAP): String Representation of

Distinguished Names, June 2006。

RFC 4515: Lightweight Directory Access Protocol (LDAP): String Representation of Search Filters, June 2006。

RFC 4522: Lightweight Directory Access Protocol (LDAP): The Binary Encoding Option, June 2006。

RFC 4523: Lightweight Directory Access Protocol (LDAP) Schema Definitions for X.509 Certificates, June 2006。

RFC 4630: Update to DirectoryString Processing in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, 2006-08, 被 RFC 5280 替代。

RFC 4683: Internet X.509 Public Key Infrastructure Subject Identification Method (SIM), 2006-10。

RFC 4985: Internet X.509 Public Key Infrastructure Subject Alternative Name for Expression of Service Name, 2007-08。

RFC 5019: The Lightweight Online Certificate Status Protocol (OCSP) Profile for High-Volume Environments, 2007-09。

RFC 5055: Server-Based Certificate Validation Protocol (SCVP), 2007-12。

RFC 5272: Certificate Management over CMS (CMC), 2008-06, 被 RFC 6402 更新。

RFC 5273: Certificate Management over CMS (CMC): Transport Protocols, 2008-06, 被 RFC 6402 更新。

RFC 5274: Certificate Management Messages over CMS (CMC): Compliance Requirements, 2008-06, 被 RFC 6402 更新。

RFC 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, 2008-05, 被 RFC 6818 更新。

RFC 5480: Elliptic Curve Cryptography Subject Public Key Information, 2009-03。

RFC 5636: Traceable Anonymous Certificate, 2009-08。

RFC 5697: Other Certificates Extension, 2009-11。

RFC 5755: An Internet Attribute Certificate Profile for Authorization, 2010-01。

RFC 5756: Updates for RSAES-OAEP and RSASSA-PSS Algorithm Parameters, 2010-01。

RFC 5758: Internet X.509 Public Key Infrastructure: Additional Algorithms and Identifiers for DSA and ECDSA, 2010-01。

RFC 5816: ESSCertIDv2 Update for RFC 3161, 2010-04。

RFC 5877: The application/pkix-attr-cert Media Type for Attribute Certificates, 2010-05。

RFC 5912: New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX), 2010-06, 被 RFC 6960 更新。

RFC 5913: Clearance Attribute and Authority Clearance Constraints Certificate Extension, 2010-06。

RFC 6025: ASN.1 Translation, 2010-10。

RFC 6170: Internet X.509 Public Key Infrastructure -- Certificate Image, 2011-05。

RFC 6277: Online Certificate Status Protocol Algorithm Agility, 2011-06, 被 RFC 6960 替代。

RFC 6402: Certificate Management over CMS (CMC) Updates, 2011-11。

RFC 6664: S/MIME Capabilities for Public Key Definitions, 2012-07。

RFC 6712: Internet X.509 Public Key Infrastructure—HTTP Transfer for the Certificate Management Protocol (CMP), 2012-09。

RFC 6818: Updates to the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, 2013-01。

RFC 6844: DNS Certification Authority Authorization (CAA) Resource Record, 2013-01。

RFC 6960: X.509 Internet Public Key Infrastructure Online Certificate Status Protocol—OCSP, 2013-06。

RFC 7030: Enrollment over Secure Transport, 2013-10。

4. 其他标准规范

ITU-T X.208 Specification of Abstract Syntax Notation One (ASN.1)。

ITU-T X.690 Information Technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules(CER) and Distinguished Encoding Rules (DER)。

Specification for Integrated Circuit(s) Cards Interface Devices (CCID)。

Interoperability Specification for ICCs and Personal Computer System (PC/SC)。

Microsoft Cryptographic Service Provider。

Java Cryptography Architecture (JCA)。

Java Cryptography Extension (JCE)。

2.7 PKI/信任模型

如同现实生活中的二代身份证一样，在网络世界中，通过验证对方数字证书的合法性就可确认对方身份，从而帮助互不认识或互不相信的交易双方建立信任关系。显然，PKI 体系中已经蕴含“信任”的概念。

什么是“信任”呢？在 ITU X.509 规范中，“信任”定义为“当实体 A 认为实体 B 的行为与 A 所预期的完全一致时，则称实体 A 信任实体 B”。PKI 体系中，用户因为信任 CA 中心，所以信任该 CA 中心签发的数字证书，本质上是实现了一种“信任传递”关系。需要说明的是，用户信任 CA 中心是指，用户相信该 CA 中心的所有数字证书管理行为均符合政策法规、运营规范和信息安全等要求，不会出现伪造数字证书、数字证书中信息不正确、数字证书对应的私钥不安全、数字证书作废后不及时发布等行为；用户信任数字证书，是指用户相信该数字证书持有人的身份是真实有效的，并不等于用户信任该数字证书的持有人。

从信任 CA 中心到信任其签发的用户数字证书，这种信任传递关系的前提是，该用户数字证书必须通过以下四个方面的合法性验证：

- (1) 验证该数字证书是否伪造。使用 CA 证书中公钥即可脱机验证。
- (2) 验证该数字证书中信息是否正确。由 CA 中心在签发数字证书时已保证。
- (3) 验证该数字证书是否与持证人一致。可要求持证人使用私钥对特定数据进行加密或

签名, 然后使用数字证书中的公钥来解密该数据或验签, 从而可验证持证人是否持有与数字证书中公钥对应的私钥。

(4) 验证该数字证书是否在黑名单上。通过 CRL 或 OCSP 方式可实现。

为方便理解信任传递关系, PKI 引入“信任模型”, 用于描述和分析同一 CA 管理域内部或不同 CA 管理域之间信任关系的建立和传递过程。PKI 信任模型中, CA 中心是信任的产生来源或信任起点, 称作信任锚, 而 X.509 数字证书是信任的表达和传递工具。从信任锚到数字证书, 可以构建一条信任关系传递的路径, 称作认证路径、证书路径或信任链。信任链越长, 信任传递过程中验证次数越多, 复杂度越高。图 2-4 显示了 Web 浏览器中的证书路径。

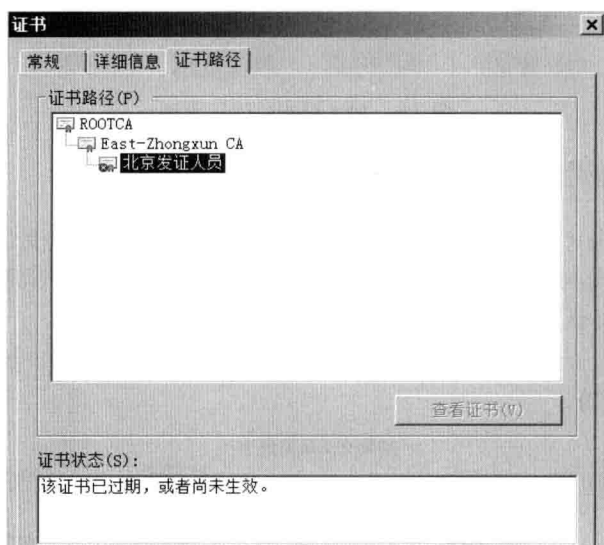


图 2-4 Web 浏览器中证书路径显示

PKI 信任模型可分为以下几类。

2.7.1 根 CA 信任模型

根 CA 信任模型, 也称作严格层次信任模型。该信任模型下, CA 中心可以分为多级, 用户证书由 CA 中心签发, 各级 CA 中心之间呈现严格的层次关系, 最上级 CA 中心只有一个, 称作根 CA, 其他 CA 称作子 CA。根 CA 的数字证书由自己签发, 属于自签名证书, 子 CA 的数字证书由上级 CA 签发。信任锚可以是根 CA, 也可以是子 CA。

如图 2-5 所示, 根 CA 签发子 CA1 证书和子 CA2 证书, 子 CA2 签发子 CA3 证书和子 CA4 证书, 子 CA1 签发用户 A 证书, 子 CA3 签发用户 B 证书和用户 C 证书, 子 CA4 签发用户 D 证书。

用户 X 的信任锚为根 CA, 因此它可信任子 CA1, 从而信任用户 A 证书。于是, 从用户 X 的角度, 用户 A 证书的信任链为: 根 CA→子 CA1→用户 A 证书。

用户 Y 的信任锚为子 CA2, 因此它可信任子 CA4, 从而信任用户 D 证书。于是, 从用户 Y 的角度, 用户 D 证书的信任链为: 子 CA2→子 CA4→用户 D 证书。

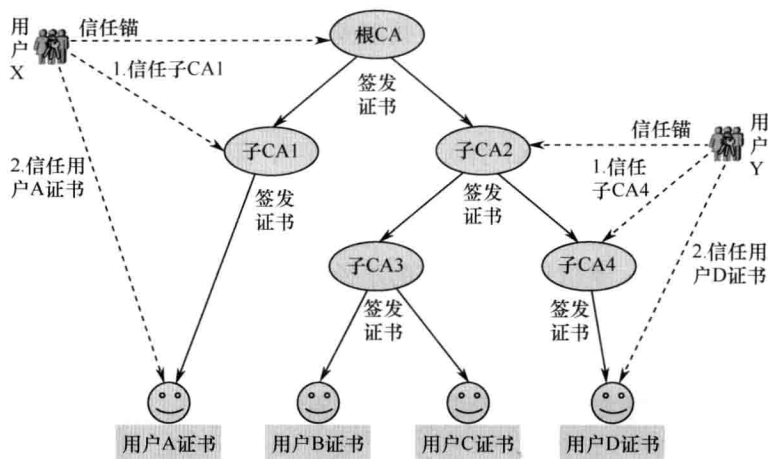


图 2-5 PKI 根 CA 信任模型

2.7.2 交叉认证信任模型

该信任模型下，根 CA 之间可以互相签发交叉认证证书，该交叉认证证书等同于子 CA 证书。在不增加信任锚的前提下，就可将信任关系传递到其他 CA 管理域。一般情况下，只有根 CA 之间签发交叉认证证书，子 CA 之间不签发交叉认证证书。

如图 2-6 所示，根 CA1 给根 CA2 签发交叉认证证书根 CA2(1)，根 CA2 给根 CA1 签发交叉认证证书根 CA1(2)。

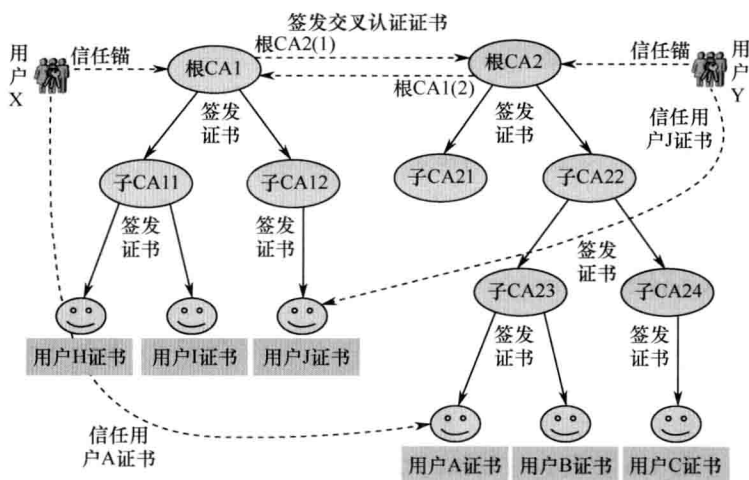


图 2-6 PKI 交叉认证信任模型

用户 X 的信任锚为根 CA1，因此它可信任根 CA2 管理域内的用户 A 证书。于是，从用户 X 的角度，用户 A 证书的信任链为：根 CA1→根 CA2(1)→根 CA2→子 CA22→子 CA23→用户 A 证书。

用户 Y 的信任锚为根 CA2，因此它可信任根 CA1 管理域内的用户 J 证书。于是，从用

用户 Y 的角度，用户 J 证书的信任链为：根 CA2→根 CA1(2)→根 CA1→子 CA12→用户 J 证书。

交叉认证信任模型中，当有 N 个根 CA 时，最多需要签发 $N(N-1)$ 个交叉认证证书。

2.7.3 桥 CA 信任模型

该信任模型下，引入独立的桥 CA 中心，等同于虚拟的根 CA。所有根 CA 与桥 CA 之间互相签发交叉认证证书。在不增加信任锚的前提下，就可将信任关系传递到其他 CA 管理域。

如图 2-7 所示，根 CA1 与桥 CA 之间的交叉认证证书为：根 CA1(q) 和桥 CA(1)；根 CA2 与桥 CA 之间的交叉认证证书为：根 CA2(q) 和桥 CA(2)。

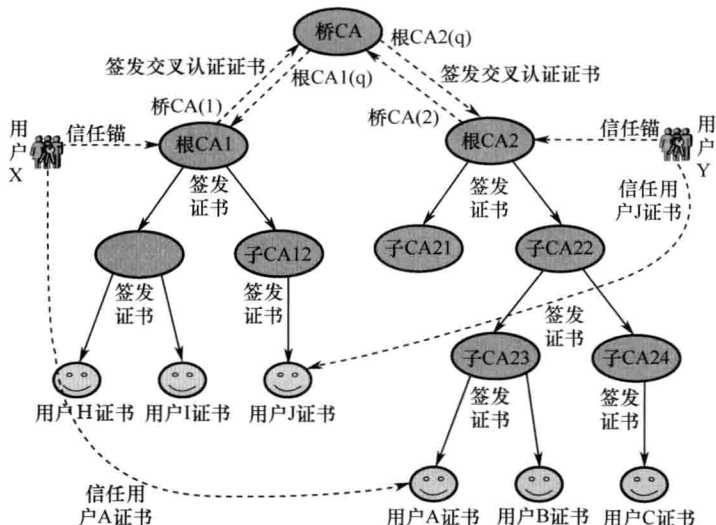


图 2-7 PKI 桥 CA 信任模型

用户 X 的信任锚为根 CA1，因此它可信任根 CA2 管理域内的用户 A 证书。于是，从用户 X 的角度，用户 A 证书的信任链为：根 CA1→桥 CA(1)→桥 CA→根 CA2(q)→根 CA2→子 CA22→子 CA23→用户 A 证书。

用户 Y 的信任锚为根 CA2，因此它可信任根 CA1 管理域内的用户 J 证书。于是，从用户 Y 的角度，用户 J 证书的信任链为：根 CA2→桥 CA(2)→桥 CA→根 CA1(q)→根 CA1→子 CA12→用户 J 证书。

桥 CA 信任模型中，当有 N 个根 CA 时，最多只需要签发 $2N$ 个交叉认证证书。

2.7.4 信任列表信任模型

该信任模型下，用户可以拥有多个信任锚，如图 2-8 所示。

用户 X 的信任锚为根 CA1 和子 CA22，因此它可信任根 CA1 管理域内的用户 H 证书，也可信任子 CA22 管理域内的用户 A 证书。于是，从用户 X 的角度，用户 H 证书的信任链为：根 CA1→子 CA11→用户 H 证书；用户 A 证书的信任链为：子 CA22→子 CA23→用户 A 证书。

用户 Y 的信任锚为根 CA2 和根 CA1，因此它可信任根 CA2 管理域内的用户 C 证书，也可信任根 CA1 管理域内的用户 J 证书。于是，从用户 Y 的角度，用户 C 证书的信任链为：

根 CA2→子 CA22→子 CA24→用户 C 证书；用户 J 证书的信任链为：根 CA1→子 CA12→用户 J 证书。

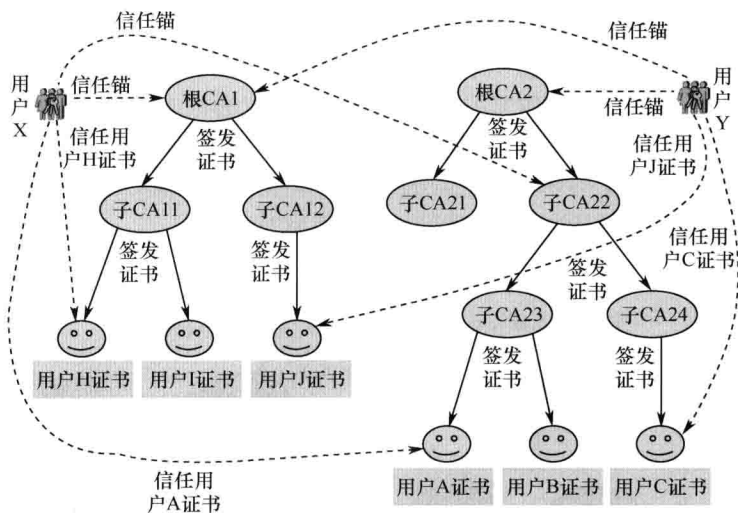


图 2-8 PKI 信任列表信任模型

Web 浏览器属于典型的信任列表信任模型，已经内置了多个信任锚，如图 2-9 所示。



图 2-9 Web 浏览器已内置多个信任锚

第3章 其他非对称密钥管理体系

根据数字证书格式及密钥管理方式的不同，PKI 也包括多种模式，如 X.509 模式、PGP 模式、IBE/CPK 模式、EMV 模式等。X.509 模式的 PKI 也称作 PKIX。由于 X.509 标准已经成为数字证书格式的事实标准，因此大部分情况下 PKI 特指 PKIX。如非特殊说明，本书中 PKI 均指 PKIX。

3.1 PGP

1. PGP 简介

PGP 是 Pretty Good Privacy 的简称，字面含义是完美隐私。PGP 实际是一个基于公钥算法的加密软件，提供文件安全保护、安全电子邮件、VPN 安全通信等功能。文件安全保护主要包括文件或文件夹加密、硬盘或虚拟磁盘加密、文件粉碎或擦除等功能。安全电子邮件主要包括邮件内容加密、邮件发送者身份认证等功能。

PGP 诞生于 1991 年，由原作者 Philip Zimmermann 发表，属于开源且免费软件。由于受 Symantec 公司收购影响，PGP 从 10.0.2 版本开始将不再独立发布，而是以安全插件形式集成于 Symantec 公司的安全产品中，如 Norton。

2. PGP 密钥格式：密钥环

PGP 为每个节点（或用户）提供一对数据结构，一个用于存放本节点（或用户）自身的公钥对，另一个用于存放本节点知道的其他用户的公钥，即私钥环（Private Key Ring）和公钥环（Public Key Ring），如图 3-1 所示。

私钥环				
Timestamp	Key ID*	Public Key	Encrypted Private Key	User ID*
⋮	⋮	⋮	⋮	⋮
T _i	$KU_i \bmod 2^{64}$	KU_i	$E_{H(P_i)}[KR_i]$	User i
⋮	⋮	⋮	⋮	⋮

公钥环							
Timestamp	Key ID*	Public Key	Owner Trust	User ID*	Key Legitimacy	Signature(s)	Signature Trust(s)
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
T _i	$KU_i \bmod 2^{64}$	KU_i	trust_flag _i	User i	trust_flag _i		
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

*=field used to index table

图 3-1 PGP 私钥环和公钥环

其中, User ID 表示用户名称, 通常用邮件地址表示; Private Key 表示私钥, 使用口令加密存储, 保存于 Encrypted Private Key 中; Public Key 表示公钥; Key ID 表示密钥唯一标识; Timestamp 表示时间戳。私钥环和公钥环可以用 User ID 或 Key ID 进行索引。

3. PGP 密钥管理

为有效解决好密钥与用户映射关系的难题, PGP 引入了公钥介绍机制, 基本原理是:

(1) 公钥信任级别分为: 最高信任 (ultimate trust)、完全信任 (complete trusted)、接近信任 (marginally trusted)、不信任 (untrusted)、不认识 (unknown)。

(2) 每个用户可以使用自己的私钥对他人的公钥进行签名。每个公钥可以拥有多个他人的签名。

(3) 向公钥环新增公钥时, 如果能完全确认该公钥的拥有者, 则该公钥的信任级别设置为 Ultimate Trust。

(4) 如果新增公钥拥有多个签名, 则通过公钥环中这些签名对应公钥的信任级别, 来确定该新公钥的信任级别。

PGP 密钥管理实际是模拟现实社会中人们的交往关系。公钥可以由单个朋友或多个朋友推荐, 根据推荐者的信任程度自动将公钥分为不同的信任级别, 最后由用户参考决定对该公钥的信任程度。

4. PGP 密钥应用

PGP 密钥应用主要包括六种基本功能或服务: 认证、保密、认证与保密、压缩、邮件兼容性、分段, 如表 3-1 所示。

表 3-1 PGP 基本功能或服务

基本功能	使用算法	说明
认证	RSA SHA	将消息按照 SHA1 算法产生消息摘要, 使用发送者私钥对消息摘要按照 RSA 算法加密(签名)。将消息明文、签名一起发送
保密	RSA CAST、IDEA、3DES	按照算法 CAST-128、IDEA 或 3DES 产生会话密码后将消息加密, 使用接收方公钥按照 RSA 算法加密会话密钥。将消息密文、会话密钥密文一起发送
认证+保密	RSA SHA CAST、IDEA、3DES	将消息按照 SHA1 算法产生消息摘要, 使用发送者私钥对消息摘要按照 RSA 算法加密(签名); 按照算法 CAST-128、IDEA 或 3DES 产生会话密码后将消息与签名加密, 使用接收方公钥按照 RSA 算法加密会话密钥。将消息与签名密文、会话密钥密文一起发送
压缩	ZIP	消息在传送或存储或加密前用 ZIP 压缩
邮件兼容性	Base 64	为了对电子邮件应用提供透明性, 使用 Base 64 算法将消息明文或密文转换成 ASCII 字符串
分段	无	为了符合最大消息尺寸限制, 对消息进行分段和重新组装

3.2 EMV

1. EMV 简介

EMV 组织由 EUROPAY、MASTERCARD、VISA 国际组织共同成立, 负责管理、维护和修订 EMV 标准, 制定相关检测案例, 以保证其可在全球范围内通用。2002 年 EUROPAY

与 MASTERCARD 合并,2005 年 JCB 购入 EMV Co.的 1/3 股份,2009 年美国运通也加入 EMV 组织,拥有 1/4 股份。

EMV 标准是关于智能卡及读卡终端用于银行卡支付的规范,着眼于取代磁条卡,实现全球范围的跨国界、跨厂商、跨金融机构的互操作,并提供一卡多功能应用的基础。EMV 建立在 ISO 7816 系列标准基础之上,定义了智能卡及终端之间的通信规范、卡以及持卡人的授权方法以及卡与终端的风险管理框架,从技术上分为两个层次,设备供应商必须取得这两个层次的测试及认可。EMV 标准主要包括:EMV96 (EMV 3.1.1)、EMV2000 (EMV 4.0)、EMV4.1、CCD (EMV 通用核心定义)、CPA (EMV 通用支付规范)、EMV 4.2 等。

EMV 迁移是按照 EMV 标准,在发卡、业务流程、安全控管、受理市场、信息转接等多个环节实施推进银行磁条卡向芯片卡技术的升级,即把现在使用磁条的银行卡改换成使用 IC 卡的银行卡。随着信息技术、微电子技术和 EMV 标准的完善及国际 EMV 迁移计划的实施,而且银行磁条卡存在伪卡、盗卡欺诈等方面的安全隐患,银行磁条卡向 IC 卡的迁移是必然的发展趋势。

2. EMV 密码应用需求

银行 IC 卡借记/贷记业务的密码技术需求是由银行卡组织制定的相关 IC 卡标准和 IC 卡芯片技术这两方面决定的。与银行借贷记磁条卡相比,发行银行 IC 卡在技术上要复杂得多。由于每张 IC 卡片中都保存了卡片的唯一密钥和由发卡行签发的静态数据或证书,因而要发行银行 IC 卡片的发卡行必须建设一整套密钥管理体系,同时还需要对相关系统进行改造,整体密码应用需求包括以下几个方面。

(1) 密钥管理体系和技术需求。

需要建立一整套银行 IC 卡借贷记应用相关的对称和非对称密钥管理体系结构、安全合理的密钥交换技术等密码需求。

(2) 脱机认证密码应用需求。

建立可实现的脱机认证流程和密码算法,通过数字证书和对称算法的交易证书方式实现脱机业务的合法性和可验证性。

(3) 数据保护及完整性密码应用需求。

密码技术应用于银行 IC 卡借贷记业务中的数据传输保护和完整性验证等,如 PIN 转加密、验证、数据加密、MAC 计算和验证等。

(4) 联机发卡行认证授权密码应用需求。

通过专用密码技术实现银行 IC 卡借贷记和电子现金应用的发卡行授权机制,替代原有银行磁条卡借贷记的卡号密码验证和信用卡 CVV 验证等方式的发卡行授权机制。

3. EMV 非对称密钥管理体系

非对称密钥管理体系主要用于支撑 IC 卡脱机业务安全和认证。EMV 非对称密钥采用两级密钥管理体系,如图 3-2 所示,包括根 CA 系统和发卡行 CA 系统。

(1) 根 CA 系统。

该系统主要用于生成根 CA 证书;接受发卡行证书申请,为发卡行签发公钥证书;向收单行发布根 CA 公钥,通过收单行将根 CA 公钥分发到受理终端。

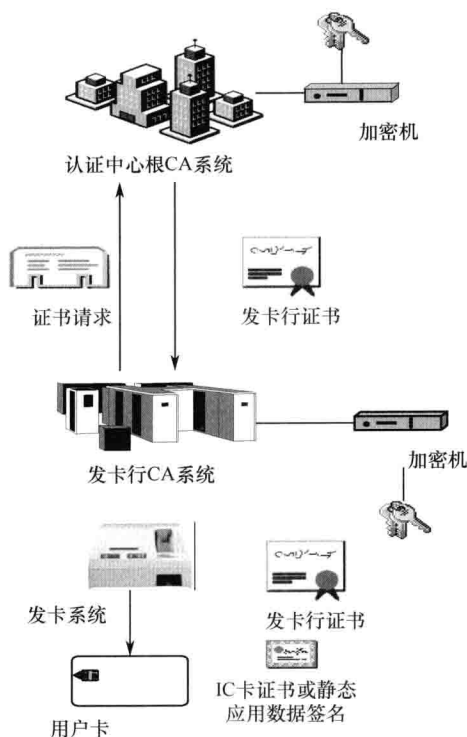


图 3-2 EMV 两级非对称密钥管理体系结构

(2) 发卡行 CA 系统。

发卡行借贷记应用 CA 系统是整个系统的安全核心系统，主要负责完成发卡行证书的申請、管理、IC 卡应用相关的密钥（包括应用密钥、卡片个人化交换主密钥等）管理，同时负责分发各类密钥到卡片制造商、卡片个人化中心及业务前置交易加密机等。

4. 密钥种类

EMV 非对称密钥管理涉及的密钥种类主要有根 CA 公私钥对、发卡行公私钥对和 IC 卡公私钥对三种，如表 3-2 所示。

表 3-2 非对称密钥列表

类 型	功 能
根 CA 公钥	由认证中心产生，以公钥证书文件形式下发。发卡行用于验证发卡证书有效性
根 CA 私钥	由认证中心产生，存储在加密机中。用于签发发卡行公钥证书
发卡行公钥	由发卡行产生，并经过 CA 中心签发后形成发卡行公钥证书。用于发卡时装载到 IC 卡中
发卡行私钥	由发卡行产生，并通过加密机密钥加密后存储在主机数据库中。用于签发 IC 卡静态数据签名及 IC 卡公钥证书
IC 卡公钥	采用 DDA 认证方式的卡片需要此密钥，由发卡行私钥签发形成 IC 卡公钥证书存储在 IC 卡
IC 卡私钥	采用 DDA 认证方式的卡片需要此密钥，用于 IC 卡与终端进行 DDA 认证

5. IC 卡证书与网上银行 PKI 证书的差异性

银行 IC 卡证书和网上银行 PKI 证书体系有很大差异，一种是符合 PKI 标准的 X.509 证

书；一种是符合金融 IC 卡标准的证书（EMV 标准的外卡与此方式类似）。以下对这两种证书及体系进行分析比较。

（1）证书格式。

① 证书的数据组织格式完全不同。PKI 证书采用 DER 编码方式，银行 IC 卡证书采用固定长度方式。

② 银行 IC 卡证书数据为密文数据，需解密后才能解析；X.509 证书数据为明文。

③ 银行 IC 卡证书数据中包括的关键数据项与 X.509 证书完全不同，并且证书相关数据项数量比较多，信息也比较复杂，如下：

PKI X.509 证书关键数据项包括：版本号、序列号、签名算法标识符、认证机构、有效期限、主题信息、认证机构的数字签名、公钥信息等。

银行 IC 卡证书关键数据项包括：证书格式标识、应用主账号、应用主账号序列号、证书失效日期、证书序列号、注册的应用提供商标识、认证中心公钥索引、应用交互特征 AIP、应用生效日期、应用失效日期、应用用途控制 AUC、持卡人验证方法（CVM）列表、发卡行公钥证书、发卡行公钥指数、发卡行行为代码（缺省、拒绝、联机）、发卡行国家代码等。

（2）发证方式。

网银 PKI 证书与 IC 卡证书发证方式如表 3-3 所示。

表 3-3 网银证书与 IC 卡证书发证方式的比较

比较项	网银证书 X.509	银行 IC 卡证书
外围系统支持	不需要外围系统支持	需要申请和接收卡组织支付系统根 CA 系统签发的发卡行证书。 需要银行核心系统提供账户账号
预制证书	X.509 证书中的“主题信息”项中必须包括持卡人姓名等个人信息。实现预制证书业务会导致证书中持卡人信息项无效，证书绑定制卡人后，通过后台系统进行关联。 证书存储介质采用 USB Key，没有配套的批量设备，只能形成证书数据文件，通过柜面交易写入设备	与持卡人信息无关，但是需要银行核心系统提供账号段。 证书存储介质采用 IC 卡，由批量设备将预制证书写入 IC 卡内，形成预制卡。 通过柜面交易绑定个人后，将制卡人信息写入 IC 卡
柜面在线发证	由 USB Key 产生公私钥对，公钥及相关信息通过 RA 系统等提交后台 CA 系统签发证书	不支持
预约卡	不适用，没有此类业务	支持贷记 IC 卡的预约制卡业务
签名及摘要算法	RSA，SHA-1	RSA，SHA-1
证书密钥对	两种方式： （1）USB Key 生成 （2）密码机生成	只支持密码机生成
证书有效期	与证书密钥对无关，由银行网银 CA 系统定义	与证书密钥对相关，由银行卡组织（中国银联、EMV）定义
证书唯一性	绑定个人。每个用户只能拥有一张网银证书	绑定账户账号。每个账户只能有一张证书，每个用户可以拥有多种证书

(3) 交易业务。

对于基于网银证书和银行 IC 卡证书开展的银行金融业务来说, 差异比较大。网银证书主要为互联网上银行业务服务; 银行 IC 卡证书主要为银行卡借记、贷记业务服务, 与原有的银行磁条卡的业务相同, 如表 3-4 所示。

表 3-4 网银证书与 IC 卡证书交易业务的比较

比较项	网银证书 X.509	银行 IC 卡证书
证书发布	支持	不支持
黑名单发布	支持	不支持
柜面业务	不适用, 不需要证书认证	需要证书认证
互联网业务	支持, 需要证书认证, 并实现操作签名和数据加密等功能	不支持
数据加密	支持	不支持
POS 机交易	不支持	支持

具体技术细节请参考《商业银行密码技术应用》“第五章 借记/贷记/电子现金密码应用体系”。

第二部分

PKI 技术基础

第4章 ASN.1 及其编码规则

4.1 ASN.1

ASN.1 是 Abstract Syntax Notation One (抽象文法描述语言) 的缩写。ASN.1 对于大多数人来讲似乎很陌生, 然而事实上它就应用在我们生活的周围。目前北美、欧洲和日本等地使用的移动电话, 都是基于 TCAP 消息协议的。TCAP 消息协议中的消息是采用 ASN.1 描述的, 使用了 BER (Basic Encoding Rules) 编码规则, 共同实现了移动电话的呼叫。可以说, 在一部移动电话与另外一部电话之间通信时, 是 ASN.1 协助实现了两部话机之间的呼叫。后来制定的有关地对空和地对地等通信协议都是使用 ASN.1 描述的, 并采用 PER (Packed Encoding Rules) 编码规则。除此之外, ASN.1 和编码规则还被联邦快递用于大量地传输信息; 还有许多大公司如 HP、IBM、SUN 等, 使用 ASN.1 描述其打印机打印作业管理的标准接口。有名的简单网络管理协议 (Simple Network Management Protocol, SNMP) 就是用 ASN.1 对所有数据进行描述的。

ASN.1 是一种对分布计算机系统之间交换的数据消息进行抽象描述的规范化语言。以前, ASN.1 只用于撰写国际通用标准。然而, 随着 ASN.1 软件工具的出现, ASN.1 已经用于生成应用程序编程语言代码, 成为各种消息系统应用的核心。现在 ASN.1 成为描述通信协议的标准文法, 而且对通信协议的描述, 不用再区分通信程序实现的编程语言和通信数据的原始表示, 也不用再区分应用系统的复杂或简单。

总之, ASN.1 是一种国际标准, 它为抽象数据结构的描述说明定义了一种记法。ASN.1 使用抽象法对各种编程语言定义的数据类型进行了重新定义, 将所有数据分为两大类: 基本类型 (如整型、布尔类型、字符串类型和比特串类型等) 和结构类型 (如结构、链表和选择类型等)。使用抽象法描述的系统让设计者可以只关心系统的某部分, 而不必关心系统中的某一部分功能如何实现或者所代表的内容。抽象法作为软件开发管理的关键, 已经为越来越多的软件设计开发人员所认可。对于某个开放使用的应用层程序, 抽象法可以简化程序的说明, 同时可以将程序的执行过程及其相关部分表述成抽象语言; 当上一级程序使用下一级应用程序时也使用抽象法简化其过程, 抽象法在越来越多的软件中得到应用。简单地说, 抽象法通过抽象文法描述将数据结构进行抽象化描述, 使用抽象文法描述的应用消息作为该应用的消息协议, 如果该应用涉及多级应用程序, 每一级应用只构造或解释与该级应用相关的信息, 而不必关心上下级应用的信息。

ASN.1 作为抽象描述文法, 将现有的数据类型抽象描述成近 20 种数据类型。这些数据类型主要分为两大类: 基本类型和结构类型。

基本类型又称为原子类型, 是构成其他结构类型的成员类型, 主要包括:

- ① 布尔类型: BOOLEAN
- ② 整型: INTEGER
- ③ 比特串: BIT STRING

- ④ 字节串: OCTET STRING
- ⑤ 空: NULL
- ⑥ 对象标识: OBJECT IDENTIFIER
- ⑦ 可打印字符串: PrintableString
- ⑧ IA5 字符串: IA5String
- ⑨ 可见字符串: VisibleString
- ⑩ 数字字符串: NumericString
- ⑪ BMP 字符串: BMPString
- ⑫ 枚举类型: Enumerated
- ⑬ UTC 时间类型: UTCTime
- ⑭ Generalized 时间类型: GeneralizedTime
- ⑮ 任意类型: ANY

其中,布尔类型是任意取值只为 0 或者 1 的数据类型;整型是任意的整数数据类型;比特串是以比特为单位任意的字符串(0, 1 串);字节串是以字节为单位的任意字符串;空类型是 NULL;对象标识是使用一串数字标识一个实体,例如一种算法或一种属性;可打印字符串是任意可打印字符组成的字符串;IA5 字符串是任意 ASCII 字符组成的字符串;数字字符串是字符 0 到 9 任意组成的字符串;BMP 字符串是使用两个字节表示一个字节数据的字符串;枚举类型与编程语言中描述的枚举类型一样;UTC 时间类型是格林尼治时间的数据类型;Generalized 时间类型是表示本地时间的数据类型;任意类型是对使用编码规则编码生成的信息定义的数据类型。

结构类型又称为复合类型,主要包含

- ① 有序成员固定结构: SEQUENCE
- ② 无序成员固定结构: SET
- ③ 有序成员待定结构: SEQUENCE OF
- ④ 无序成员待定结构: SET OF
- ⑤ 选择类型: CHOICE

其中,有序成员固定结构是指使用前已确定数据成员的个数和顺序的结构体类型;无序成员固定结构是指使用前已确定数据成员的个数,但未确定数据成员顺序的结构体类型;有序成员待定结构是指使用前已确定数据成员数据,使用时才确定数据成员的个数的结构体链表类型;无序成员待定结构是指使用时才确定数据成员的个数和顺序的结构体链表类型;选择类型是由几种数据类型的数据成员构成的共同体类型。

通过 ASN.1 抽象定义后的数据类型几乎概括了现实世界中存在的所有数据类型,具有相当的通用性。在制定应用系统消息协议时就使用这些数据类型描述消息结构,屏蔽了各种编程语言自身的数据类型,提高了消息通用性。同时,由于使用了这些抽象数据类型描述消息协议,还克服了原编程语言描述消息结构的许多弊病。例如,在使用编程语言描述的消息协议中,协议设计人员为了使协议具有一定的扩展性,需要在一些数据结构中保留字段,由于定义保留字段时无法确定以后要扩展的数据类型,在使用编程语言描述协议时,都将保留字段定义成字符串类型。然而,在后来的使用过程中,可能需要扩展的数据类型是结构体,或者扩展的数据项增多,原来定义的保留字段的个数不足时,造成一个保留字

段中存放多个数据或多个数据结构的组合。为消息双方实现消息处理增加了大量的代码，用于对保留字段中多个数据成员的解析，同时，还可能修改消息协议的描述。

如果使用 ASN.1 描述这些消息协议，将使用 ASN.1 中定义的数据类型代替编程语言中的数据类型来描述消息协议中的结构。可以将保留字段定义成一个有序成员待结构类型的数据，称为扩展项集。扩展项集的成员是扩展项，扩展项是有序成员固定结构类型，扩展项可以简单包含两个成员：扩展项标识（对象标识类型）和扩展项信息（任意类型，是具体扩展数据的 DER 编码）。在消息协议使用过程中没有扩展项时，该扩展项集无须赋值。需要加入扩展项时，给扩展项集中一个扩展项赋值，即填充扩展项类型标识（表示是何种扩展项）和扩展项信息。有多个扩展项时，由于扩展项集是有序成员待结构类型的数据，可以在使用时随意加入数据成员（加入多个扩展项）。这样无须增加对扩展项信息组合和解析的代码，双方也不必协商扩展项在保留字段中如何组合，也不受扩展项数据类型和个数的限制。同时，根据扩展项标识可以轻松获得接受方所需的扩展项。

ASN.1 抽象文法描述的优势，在应用系统消息协议中，特别是大型消息协议如电子商务体系中证书认证系统和支付系统协议等方面得到了发挥。使得 ASN.1 文法描述越来越得到系统设计和软件开发人员的一致认可，并且被更为广泛地应用。

4.2 BER（基本编码规则）与 DER（定长编码规则）

4.2.1 数据类型标识

4.2.1.1 简介

ASN.1 应用越来越广泛的主要原因之一是这种抽象文法描述与几种标准化编码规则联系在一起。这些编码规则规定了如何在非传输介质下实现 ASN.1 定义的数据类型与字节流的相互转换。因为编码规则是针对 ASN.1 描述的数据类型而制定的，因此可以称这些编码规则为 ASN.1 的编码规则。

ASN.1 的编码规则是把使用 ASN.1 语言说明的数据转化成一种标准格式的系列规则，同时，保证转换后的数据在任意操作系统中，只要使用相同编码规则的解码器就可以解码获得原始数据。可见，ASN.1 和相应的编码规则保证了数据在分布式计算机系统中传输的一致性。对同一个 ASN.1 描述的数据可以采用不同的编码规则，选择使用哪一种编码规则取决于协议设计者的设计初衷。

目前，ASN.1 的标准化的编码规则有以下几种：BER（Basic Encoding Rules），DER（Distinguished Encoding Rules），PER（Packed Encoding Rules）和 CER（Canonical Encoding Rules）等。其中，BER 是 20 世纪 80 年代初制定的，被广泛用于应用系统中，如用于互联网管理的简单网络管理协议（Simple Network Management Protocol, SNMP），用于互传电子邮件的消息处理服务（Message Handling Service, MHS），以及用于控制计算机和电话交互信息使用的 TSAPI 等。DER 是 BER 编码的一种特殊形式，它专门适用于具有安全特性的应用系统，如涉及加密技术和要求编解码信息唯一性的系统，如电子商务系统。CER 与 DER 类似，是 BER 编码的另外一种特殊形式，CER 的最大特点是对大数据实现编码，而且在数据还未完全获得之前就可以进行编码。但是由于业界认定在安全传输中最好的编码

方法是 DER 编码，所以 CER 编码未得到广泛使用。PER 是最近制定的系列编码规则，它的显著特点是使用有效的算法对数据编码，获得比 DER 更快更紧凑的数据，因此 PER 常常被应用于带宽或 CPU 饿死状态的应用系统，如空中交通管制和视听通信等领域。

4.2.1.2 数据类型标识和派生数据类型标识

1. 数据类型标识

ASN.1 的数据类型有原子类型和结构类型两类，在使用过程中根据需要还可以从原有的数据类型派生出新的类型，称为派生类型。在 ASN.1 定义的数据类型中除了 CHOICE 和 ANY 两种类型以外，其他每种类型都有一个唯一的类型标识 (Tag)，用于标识一种数据类型。由于 CHOICE 是在几种数据类型中任意选择一个数据类型的数据，因此 CHOICE 类型标识可以是被选择数据的数据类型的类型标识；ANY 是某一类型数据编码后的信息，因此无法定义其标识。

ASN.1 中数据类型的标识分为四类：

(1) 通用类 (Universal 类)，此类标识的值在所有应用中的定义相同；

(2) 应用类 (Application 类)，此类标识的值只为某一种应用定义；

(3) 私有类 (Private 类)，此类标识的值是为某些企业或公司定义；

(4) 上下文说明类 (Context-Specific 类)，此类标识的值是为某个特定的类型定义的；由于上下文中使用了相同的数据类型的数据，并且其中有可选项，为了区别上下文中相同的数据类型的具体赋值，将其中的一个数据类型通过派生的方式改变类型标识，改变后的类型标识的类型就是上下文说明类。

由于类型标识是对数据类型的唯一标识，所谓的唯一标识应该是通用类的类型标识。表 4-1 给出了常用数据类型的通用类标识。

表 4-1 常用数据类型的通用类标识

类型	标识 (十进制)	标识 (十六进制)
BOOLEAN	1	01
INTEGER	2	02
BIT STRING	3	03
OCTET STRING	4	04
NULL	5	05
OBJECT IDENTIFIER	6	06
UTF8String	12	0C
SEQUENCE 和 SEQUENCE OF	16	10
SET 和 SET OF	17	11
NumericString	18	12
PrintableString	19	13
T61String	20	14
IA5String	22	16
UTCTime	23	17
GeneralizedTime	24	18
BMPString	30	1E

2. 派生数据类型标识

在类型标识的上下文说明类中,如果某个结构类型的成员中有若干相邻的可选项成员,其中有两项成员数据相同,由于解码时无法区别是哪一個成员的类型标识,因此需要采用派生的方法改变其中一个数据成员的类型标识。派生的数据类型标识的类型是除了通用类以外的三种标识类型,一般情况是上下文说明类。派生数据类型标识的方法有两种:显式派生法和隐式派生法。

显式派生法在 ASN.1 描述中使用[[Class] Number] Explicit 作为关键字,用它声明的类型表示使用了显式派生法,派生后生成的 Class 类型的数据类型标识,其值为 Number 的数值。Class 的类型可以是通用类、应用类和私有类,但未做声明时表示是上下文说明类,通常情况下 Class 空缺。显式派生法派生的数据类型在编码过程中与原始类型数据的编码不同。对于显式派生数据类型的编码,首先使用原数据的通用类标识对数据进行编码,再使用派生数据类型的标识对原数据获得编码信息并进行二次编码。

隐式派生法在 ASN.1 描述中使用[[Class] Number] Implicit 作为关键字,用它声明的类型表示使用了隐式派生法,派生后生成的 Class 类型的数据类型标识,其值为 Number 的数值。Class 的类型可以是通用类、应用类和私有类,但未做声明时表示是上下文说明类,通常情况下 Class 空缺。隐式派生法派生的数据类型在编码过程中与原始类型数据的编码不同。隐式派生数据类型的编码是使用派生获得的数据类型标识代替原数据的通用类标识对数据进行编码,但编码规则还是原数据类型的编码规则。

显式派生法和隐式派生法与原数据编码的比较如图 4-1 所示。

显式派生法	<table><tr><td>新Tag编码值</td><td>原Tag编码的数据</td></tr></table>	新Tag编码值	原Tag编码的数据
新Tag编码值	原Tag编码的数据		
隐式派生法	<table><tr><td>新Tag替换原Tag编码的数据</td></tr></table>	新Tag替换原Tag编码的数据	
新Tag替换原Tag编码的数据			
原始数据	<table><tr><td>原Tag编码的数据</td></tr></table>	原Tag编码的数据	
原Tag编码的数据			

图 4-1 显式派生法与隐式派生法的对比

显式派生法用于任何数据类型的派生,特别适合于 CHOICE 和 ANY 类型,因为这两种类型没有自己的类型表示,使用隐式派生法无法替换原有数据类型标识。隐式派生法用于除了 ANY 类型和 CHOICE 类型以外的所有类型的派生。

4.2.2 BER 基本编码规则

BER 编码信息由以下几部分组成:

① 标识串,表示要编码的 ASN.1 类型的标识类和标识码以及使用的编码方法(是简单型还是结构型)。

② 长度串,定长型编码方法中它表示内容串的长度,非定长编码方法中它表示长度不定。

③ 内容串,简单定长型编码方法中它表示要编码类型值的具体内容,结构型编码方法中表示各个成员编码的串联。

④ 内容结束串,只有在结构非定长型编码方法中表示内容串的结束,其他方法中该串省略。

对一个 ASN.1 对象的 BER 编码有三种模式, 使用哪一种模式取决于该对象的类型和该类型数据的长度是否已知, 不同模式下编码信息中每个组成部分的编码规则不同。

4.2.2.1 基本类型定长模式

基本类型定长模式编码主要应用于基本类型和从基本类型通过隐式派生得到的类型, 同时需要已知编码值的长度。各部分的 BER 编码规则如下。

1. 标识串编码

标识串编码分为低标识编码和高标识编码两种形式。

(1) 低标识编码。

- ① 适用于类型标识值小于 30 的类型。
- ② 编码结果只有 1 个字节。
- ③ 其中第 8 位和第 7 位表示 Class 类型, 第 8 位和第 7 位的赋值规则参见表 4-2。

表 4-2 第 8 位和第 7 位的赋值规则

Class	第 8 位	第 7 位	Class	第 8 位	第 7 位
Universal	0	0	Context-Specific	1	0
Application	0	1	Private	1	1

④ 第 6 位置为 0 表示是基本类型的编码。

⑤ 第 5 位到第 1 位填充数据类型标识的值。

(2) 高标识编码。

- ① 适用于类型标识值大于 30 的类型。
- ② 编码结果至少有 2 个字节。
- ③ 除了将第 5 到第 1 位置为 1, 第一个字节与低标识编码获得的标识串的构成相同。
- ④ 第 2 个字节以后 (包含第 2 个字节) 的字节填充类型标识的值, 基数是 128, 每个字节的第 8 位作为结束标志, 除了最后一个字节外其他都置为 1。

2. 长度串编码

长度串编码分为短型和长型两种形式。

(1) 短型长度串编码。

- ① 适用于内容长度小于 127 的类型。
- ② 编码只有一个字节。
- ③ 第 8 位置为 0, 其他 7 位填充长度的值 (是内容串的长度)。

(2) 长型长度串编码。

- ① 适用于内容长度大于 127 的类型。
- ② 编码由 2~127 个字节组成。
- ③ 第 1 个字节的第 8 位置为 1, 其他 7 位填充后面填充长度值所有的字节数。
- ④ 第 2 个字节以后 (包含第 2 个字节) 的字节, 填充内容串的长度值。

3. 内容串编码

表示基本类型具体的编码值。

对于 OBJECT IDENTIFIER 类型, 假设 $\text{OID}=\text{V1.V2.V3}\cdots\text{Vn}$, 则其编码规则为: 第 1 个字节 $=40\times\text{V1}+\text{V2}$ (V1 取值范围为 0、1 或 2, V2 取值范围为 0~39); 对于 $\text{V3}\cdots\text{Vn}$ 中的每个 Vx , 以 128 为基数将 Vx 分解为多个数, 除最后一个数外其余数的最高位 (第 8 位) 置为 1 后, 成为编码后的多个字节。如 1.2.840.113549, 第 1 个字节为 $40\times 1+2=2\text{A}$ (十六进制); $840=6\times 128+48$ (十六进制), 则编码后为 2 个字节 86 48; $113549=6\times 128\times 128+77\times 128+0\text{D}$ (十六进制), 则编码后为 3 个字节 86 F7 0D。因此 1.2.840.113549 编码后为 06 06 2A 86 48 86 F7 0D。

对于 BIT STRING 类型, 编码后第 1 个字节表示填充位数或未使用位数。如 $\text{keyUsage}=11111$, 编码后为 03 F8。

对于 INTEGER 类型, 由于 DER 编码后第 1 字节第 8 位表示正负整数, 因此如果正整数第 1 字节第 8 位为 1 时, 在前填充 1 个字节 0x00。如十进制 128 (十六进制为 80), 编码后为 00 80。

对于 BOOLEAN 类型, TRUE 编码为 0xFF, FALSE 编码为 0x00。

4. 内容结束串编码

无内容结束串编码。

4.2.2.2 结构类型定长模式

结构类型定长模式编码, 主要应用于基本字符串类型、结构类型、从基本字符串类型和结构类型通过隐式派生得到的类型、从任意类型通过显式派生得到的类型, 也需要已知编码值的长度。各个部分的 BER 编码规则如下:

① 标识串编码, 与基本类型定长模式编码中的标识串编码规则基本相同, 除了将第 1 个字节的第 6 位置为 1, 表示结构类型编码。

② 长度串编码, 与基本类型定长模式编码中的长度串编码规则完全相同。

③ 内容串编码, 与基本类型定长模式编码中的内容串编码规则完全相同; 需要根据结构类型对象的不同类型而采用不同的编码。

④ 内容结束串编码, 无。

4.2.2.3 结构类型非定长模式

结构类型非定长模式编码, 主要应用于基本字符串类型、结构类型、从基本字符串类型和结构类型通过隐式派生得到的类型、从任意类型通过显式派生得到的类型, 不需要知道编码值的长度。各个部分的 BER 编码规则如下:

① 标识串编码, 与结构类型定长模式编码中的标识串编码规则完全相同。

② 长度串编码, 只有 1 个字节, 赋值为 0x80, 表示该数据编码是非定长模式编码。

③ 内容串编码, 与结构类型定长模式编码中内容串编码规则完全相同。

④ 内容结构串编码, 由 2 个字节组成, 赋值为 0x0000。

4.2.3 DER 定长编码规则

DER 编码是 ASN.1 数据类型中具有唯一编码的编码规则。DER 编码是 BER 编码的子集, 是将每一个 ASN.1 抽象对象类型表示成唯一的 1 和 0 码字符串的编码规则。这种编码

规则是为需要编码成唯一比特串的应用系统而制定的，特别是在应用安全技术的应用系统中，由于安全加密技术要求输入数据是字节流的形式，并且是与原数据唯一对应的字节流，因此需要使用 DER 编码来实现数据结构的编码。

DER 编码称为有关安全技术的应用系统的最佳选择。它基本上继承了 BER 编码规则，同样，也有三种编码方法。但为了保证编码结果的唯一性，DER 编码在 BER 编码规则的基础上又附加了一些规则：

- ① 必须使用定长模式编码。
- ② 对于内容长度小于 127 的类型值，长度串编码必须采用短型。
- ③ 对于内容长度大于 128 的类型值，长度串编码必须采用长型，同时长度串编码的字节个数必须是最少的。
- ④ 对于简单字符串类型和从简单字符串类型通过隐式派生得到的类型，必须使用基本类型定长模式编码方法。
- ⑤ 对于结构类型、从结构类型通过隐式派生得到的类型，以及从任何类型通过显示派生得到的类型，必须使用结构类型定长模式编码方法。

第5章 密码技术

5.1 密码算法

5.1.1 算法分类

按照技术特征分类，密码算法可以分为以下三类。

1. 对称算法

对称算法是指加密密钥和解密密钥相同的密码算法，又称秘密密钥算法或单密钥算法。该类算法又分为流密码算法和分组密码算法。

流密码算法又称序列密码算法，每次加密或解密一位或一字节的明文或密文。

分组密码算法将明文（密文）分成固定长度的数据块（比特块或字节块），用同一密钥和算法对每一明文（密文）块加密（解密）后得到等长的密文（明文）块，然后将密文（明文）块按照顺序组合起来最终得到明文（密文）。

常见的流密码算法包括 RC4；常见的分组密码算法包括 DES、IDEA、RC2、AES、SM4 等。

2. 非对称算法

非对称算法是指加密密钥和解密密钥不相同的密码算法，从一个密钥很难推导出另一个密钥，又称公开密钥算法或公钥算法。该算法使用一个密钥进行加密，用另一个密钥进行解密。其中加密密钥可以公开，又称公开密钥或公钥；解密密钥必须保密，又称私有密钥或私钥。

常见的非对称算法包括 RSA、DH、DSA、ECDSA、ECC、SM2 等。

3. 摘要算法

摘要算法是指把任意长的输入消息数据转化成固定长度的输出数据的一种密码算法，又称散列函数、哈希函数或杂凑函数、单向函数等。摘要算法所产生的固定长度的输出数据称作摘要值、散列值或哈希值。摘要算法没有密钥。

常见的摘要算法包括 MD5、SHA1、SM3 等。

5.1.2 对称密码算法

本节主要介绍 DES、3DES、AES、SM4 算法。

1. DES

对称算法最具代表性的是 IBM 公司提出的 DES 算法，该算法自 1977 年被美国国家标准局（NBS）颁布为商用数据加密标准后，得到了广泛应用。

DES 算法是一个分组算法，它以 64 位为分组对数据进行加解密。其密钥长度为 56 位，

由 8 个字节组成，每个字节的第 8 位用作奇偶校验。密钥可以是任意的 56 位比特块，且可在任意时间改变，其中极少数 56 位比特块被认为是弱密钥，在使用中需要避开这些弱密钥。所有的保密性依赖于密钥。

DES 算法的一般过程如图 5-1 所示。

① 初始置换：是按照固定的矩阵进行换位，此部分与密钥无关。初始置换将明文（密文）分组分成左半部分和右半部分，各 32 位长。

② 子密钥生成：外部输入的 56 位密钥（64 位中去掉 8 个校验位）通过置换和位移操作生成加密和解密需要的 16 个 48 位的子密钥。该 16 个子密钥分别用于乘积变换中的 16 轮运算。

③ 乘积变换：该过程与密钥有关，且比较复杂，是加密/解密过程的关键。该过程包括线性变换和非线性变换。该过程通过多次复杂的替代和置换方法，打乱原输入数据组，加大了非规律性，增加了系统分析的难度。乘积变换将进行 16 轮完全相同的运算，在运算过程中数据与密钥结合，最终生成 32 位长的左半部分和右半部分。

④ 末置换：是初始置换的逆变换，与密钥无关。乘积变换生成的左半部分和右半部分合在一起，经过末置换后生成 64 位的密文（明文）。

DES 算法最主要的优点是：可靠性较高、加密解密速度快、算法容易实现、通用性强。其主要的缺点是：密钥位数少、算法具有对称性、容易被穷尽法攻击、密钥管理复杂。

有些密钥会导致生成相同的 16 个子密钥，这些密钥叫作弱密钥。当密钥是全 0、全 1 或一半是全 0、一半是全 1 时，会发生这种情况。4 种弱密钥（十六进制编码，带奇偶校验位）参见表 5-1。

表 5-1 弱密钥列表

弱密钥（十六进制，带奇偶校验位）	
01 01 01 01 01 01 01 01	1F 1F 1F 1F 0E 0E 0E 0E
E0 E0 E0 E0 F1 F1 F1 F1	FE FE FE FE FE FE FE FE

有些密钥把明文加密成相同的密文，这些密钥只产生 2 个不同的子密钥，而不是生成 16 个不同的子密钥，这些密钥叫作半弱密钥，参见表 5-2。

表 5-2 半弱密钥列表

半弱密钥（十六进制，带奇偶校验位）	
01 FE 01 FE 01 FE 01 FE	FE 01 FE 01 FE 01 FE 01
1F E0 1F E0 0E F1 0E F1	E0 1F E0 1F F1 0E F1 0E
01 E0 01 E0 01 F1 01 F1	E0 01 E0 01 F1 01 F1 01
1F FE 1F FE 0E FE 0E FE	FE 1F FE 1F FE 0E FE 0E
01 1F 01 1F 01 0E 01 0E	1F 01 1F 01 0E 01 0E 01
E0 FE E0 FE F1 FE F1 FE	FE E0 FE E0 FE F1 FE F1

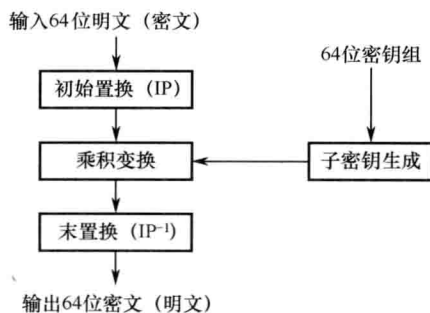


图 5-1 DES 算法的一般过程

有些密钥只产生 4 个不同的子密钥，这些密钥叫作可能的弱密钥，参见表 5-3。

表 5-3 可能的弱密钥列表

可能的弱密钥（十六进制，带奇偶校验位）	
1F 1F 01 01 0E 0E 01 01	E0 01 01 E0 F1 01 01 F1
01 1F 1F 01 01 0E 0E 01	FE 1F 01 E0 FE 0E 01 F1
1F 01 01 1F 0E 01 01 0E	FE 01 1F E0 FE 01 0E F1
01 01 1F 1F 01 01 0E 0E	E0 1F 1F E0 F1 0E 0E F1
E0 E0 01 01 F1 F1 01 01	FE 01 01 FE FE 01 01 FE
FE FE 01 01 FE FE 01 01	E0 1F 01 FE F1 0E 01 FE
FE E0 1F 01 FE F1 0E 01	E0 01 1F FE F1 01 0E FE
E0 FE 1F 01 F1 FE 0E 01	FE 1F 1F FE FE 0E 0E FE
FE E0 01 1F FE F1 01 0E	1F FE 01 E0 0E FE 01 F1
E0 FE 01 1F F1 FE 01 0E	01 FE 1F E0 01 FE 0E F1
E0 E0 1F 1F F1 F1 0E 0E	1F E0 01 FE 0E F1 01 FE
FE FE 1F 1F FE FE 0E 0E	01 E0 1F FE 01 F1 0E FE
FE 1F E0 01 FE 0E F1 01	01 01 E0 E0 01 01 F1 F1
E0 1F FE 01 F1 0E FE 01	1F 1F E0 E0 0E 0E F1 F1
FE 01 E0 1F FE 01 F1 0E	1F 01 FE E0 0E 01 FE F1
E0 01 FE 1F F1 01 FE 0E	01 1F FE E0 01 0E FE F1
01 E0 E0 01 01 F1 F1 01	1F 01 E0 FE 0E 01 F1 FE
1F FE E0 01 0E FE F1 01	01 1F E0 FE 01 0E F1 FE
1F E0 FE 01 0E F1 FE 01	01 01 FE FE 01 01 FE FE
01 FE FE 01 01 FE FE 01	1F 1F FE FE 0E 0E FE FE
1F E0 E0 1F 0E F1 F1 0E	FE FE E0 E0 FE FE F1 F1
01 FE E0 1F 01 FE F1 0E	E0 FE FE E0 F1 FE FE F1
01 E0 FE 1F 01 F1 FE 0E	FE E0 E0 FE FE F1 F1 FE
1F FE FE 1F 0E FE FE 0E	E0 E0 FE FE F1 F1 FE FE

2. 3DES

3DES 又称 Triple DES，是 DES 算法的一种变种，它使用 3 个 56 位的密钥对数据进行 3 次加密。3DES 加密/解密过程如图 5-2 所示。

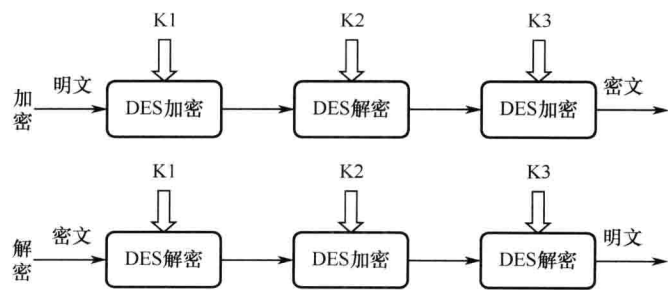


图 5-2 3DES 加密/解密过程

K1、K2、K3 决定了 3DES 算法的安全性。若 3 个密钥互不相同，则 3DES 算法的密钥长度变为 168 位，该算法又称为三倍长密钥的 3DES。如 $K1=K3$ ，则 3DES 算法的密钥长度变为 112 位，该算法又称为 2 倍长密钥的 3DES。

3. AES 简介

1997 年 4 月，美国国家标准和技术委员会 (NIST) 开始征集“高级加密标准”(AES) 算法，以便替代 DES 算法。1998 年 5 月，NIST 宣布接受 15 个新的候选算法并提请全世界密码研究界协助分析这些候选算法，包括对每个算法的安全性和效率特性进行初步检验。NIST 考察了这些初步的研究结果，并选定 MARS、RC6、Rijndael、Serpent 和 Twofish 等 5 个算法作为参加决赛的算法。经公众对决赛算法进行更进一步的分析评论，2000 年 10 月，NIST 推荐 Rijndael 作为高级加密标准(AES)，并于 2001 年 11 月 26 日发布于 FIPS PUB 197，2002 年 5 月 26 日成为正式标准。

Rijndael 是一种迭代分组密码，采用的是代替 / 置换网络 (spn)，它对一个 128 位的数据块进行加密操作。加密时，首先将输入的 128 位数据排成 4×4 的字节矩阵，然后根据不同的密钥长度，进行 10 (128 位密钥)、12 (192 位密钥) 或 14 (256 位密钥) 轮的运算。其 128 位密钥的加密流程如图 5-3 所示。

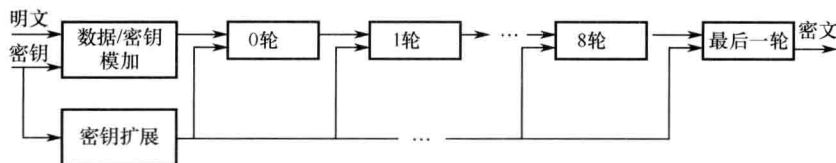


图 5-3 Rijndael 算法 128 位密钥的加密流程 (10 轮)

其中，每个轮函数由 4 层组成：第 1 层 (盒变换) 为非线性层，将一个 8×8 的 S 盒应用于每一个字节；第 2 层 (行移位变换) 和第 3 层 (列混合) 是线性混合层，将 4×4 的阵列按行位移，按列混合；在第 4 层 (加密密钥变换)，轮密钥异或到阵列的每个字节。其中，除最后一轮中没有列混合外，其他轮次均相同。密钥扩展的过程根据密钥长度的不同会有所差别。解密过程只是和加密过程稍有不同，除了 S 盒，其他过程分别是加密过程的逆运算。每一轮的流程如图 5-4 所示。

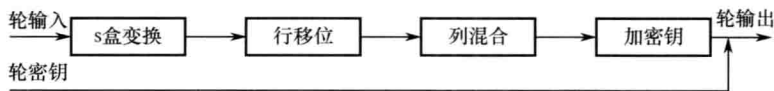


图 5-4 轮变换的过程

4. SM4 (原 SMS4)

SM4 算法是由中国国家密码管理局于 2006 年 1 月 6 日发布，在无线局域网产品中批准使用的对称密码算法。

SM4 密码算法是一个迭代分组密码算法。该算法的信息块长度为 128 位，密钥长度为 128 位。加密算法与密钥扩展算法都采用 32 轮非线性迭代结构。SM4 数据解密和数据加密的算法结构相同，只是子密钥的使用顺序相反，解密子密钥是加密子密钥的逆序。

5.1.3 非对称密码算法

本节主要介绍 RSA、ECC、SM2 算法。

1. RSA

RSA 算法是由美国三位科学家 Rivest、Shamir 和 Adleman 于 1976 年提出并在 1978 年正式发表的公开密码算法，其命名取自三位创始人名字的首字母缩写。该算法基于数论中的大数分解难题，即：根据数论，寻求两个大素数比较简单，而将它们的乘积分解开则极其困难。

该算法中，用户有两个密钥：公钥 $PK=\{e, n\}$ 和私钥 $SK=\{d, n\}$ ， n 为两个大素数 p 和 q 的乘积（素数 p 和 q 一般为 100 位以上的十进制数）， e 和 d 满足一定的关系，如只知道 e 和 n 并不能求出 d 。

(1) 加密/解密过程

若用整数 X 表示明文，用整数 Y 表示密文（ X 和 Y 均小于 n ），则加密和解密运算为：

$$\text{加密: } Y = X^e \bmod n$$

$$\text{解密: } X = Y^d \bmod n$$

(2) 密钥的产生

① 计算 n 。用户秘密选择两个大素数 p 和 q ，计算出 $n=pq$ 。 n 称为 RSA 算法的模数。明文必须能够用小于 n 的数来表示。

② 计算 $\varphi(n)$ 。用户计算出 n 的欧拉函数 $\varphi(n)=(p-1)(q-1)$ 。 $\varphi(n)$ 定义为不超过 n 并与 n 互素的数的个数。

③ 选择 e 。用户从 $[0, \varphi(n)-1]$ 中选择一个与 $\varphi(n)$ 互素的数 e 作为公开的加密指数。

④ 计算 d 。用户计算出满足公式 $ed=1 \bmod \varphi(n)$ 的 d 作为解密指数。

⑤ 得出所需要的公钥和私钥：

$$\text{公钥 } PK=\{e, n\}$$

$$\text{私钥 } SK=\{d, n\}$$

2. ECC

ECC 是椭圆曲线密码（Elliptic Curve Cryptography）的缩写。1985 年，Koblitz 和 Miller 提出基于椭圆曲线离散对数问题（ECDLP, Elliptic Curve Discrete Logarith Problem）的公钥密码体制，即椭圆曲线密码体制 ECC。它是用椭圆曲线有限群代替基于有限域上离散对数问题公钥密码中的有限循环群所得到的一类密码体制。由于在一般的椭圆曲线群（除了个别特殊的椭圆曲线以外）中没有亚指数时间算法解，所以椭圆曲线密码成了目前最流行的公钥密码体制。

椭圆曲线密码体制 ECC 的安全性基于椭圆曲线点群上离散对数问题 ECDLP 的难解性。而解 ECDLP 最有效的算法则要依靠指数时间的算法。目前对椭圆曲线密码体制最有威胁的方法是 Pollard's rho 方法和 Pohlig Hellman 方法。离散对数的求解是非常困难的，而椭圆曲线离散对数问题比有限域上离散对数问题更难求解，这意味着 ECC 能以更小的密钥长度来产生与其他公钥密码算法同等级的安全性。为了达到对称密钥 128 位的安全水平，美国国家标准技术研究所（NIST）推荐使用 3072 位的 RSA 密钥。而对 ECC 来说，256 位就可以达到同等的安全水平。

ECC 还具有以下优点:

① 计算量小, 处理速度快。虽然 RSA 可以通过选取较小公钥的方法提高公钥处理速度, 即提高加密和签名验证的速度, 使其在加密和签名验证速度上与 ECC 有可比性, 但在私钥的处理速度上 (解密和签名), ECC 远比 RSA、DSA (Digital Signature Algorithm) 快得多, 因此 ECC 总的速度比 RSA、DSA 要快得多。

② 存储空间占用少。ECC 的密钥尺寸和系统参数与 RSA、DSA 相比要小得多, 意味着它所占的存储空间要小得多。

③ 带宽要求低。对长消息进行加、解密时, ECC 与 DSA/RSA 密码算法具有相同的带宽要求, 但应用于短消息时 ECC 带宽要求却低得多。而公钥密码算法多用于短消息 (如用于数字签名和密钥交换), 带宽要求低使 ECC 在无线网络领域具有广泛的应用前景。

ECC 算法的基本原理如下。

(1) 有限域 F_p 与椭圆曲线

有限域 F_p 是由小于素数 p 的非负整数组成的集合 $\{0, 1, 2, \dots, p-1\}$, 其上的运算是模 p 的算术运算。

F_p 上的椭圆曲线是满足方程 $y^2 = x^3 + ax + b \bmod p$ 的 F_p 上的点 (x, y) 组成的集合, 其中常量 a 和 b 也是 F_p 中的元素。

为了详细描述一条 F_p 上的椭圆曲线, 应该给出如下参数:

- ① 素数 p 的值, 为满足 ECC 的安全性要求, p 应该为 160 以上位长的素数;
- ② 常数 a 和 b 的值, 其中 a 可以取 -3 以提高点运算效率;
- ③ 椭圆曲线上的一个基点 G (也称为生成元);
- ④ 基点 G 的阶为 n , 一般情况下要求 n 为素数且等于椭圆曲线上的点数。

(2) ECC 密钥对

对于给定的椭圆曲线参数 $\{p, a, b, G, n\}$, ECC 的私钥 d 为满足 $1 < d < n$ 的随机数, 相应的公钥 P 为 $P = dG$ 。为了满足安全性要求, d 必须通过随机或强伪随机数发生器产生。

(3) ECC 签名机制

ECDSA (Elliptic Curve Digital Signature Algorithm) 是典型的 ECC 签名机制。

(4) ECC 加密机制

ECIES (Elliptic Curve Integrated Encryption Scheme) 是典型的 ECC 加密机制, 请参见 ISO/IEC 18033-2。

3. SM2

SM2 算法是由国家密码管理局于 2012 年 12 月 17 日发布的椭圆曲线公钥密码算法, 主要满足电子认证服务系统等应用需求。

该算法的主要参数如下:

- ① 推荐使用素数域 256 位椭圆曲线。
- ② 椭圆曲线方程: $y^2 = x^3 + ax + b$ 。
- ③ 曲线参数:

```
p=FFFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFF
a=FFFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFF
```

b=28E9FA9E 9D9F5E34 4D5A9E4B CF6509A7 F39789F5 15AB8F92 DDBCBD41 4D940E93
n=FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 7203DF6B 21C6052B 53BBF409 39D54123
Gx=32C4AE2C 1F198119 5F990446 6A39C994 8FE30BBF F2660BE1 715A4589 334C74C7
Gy=BC3736A2 F4F6779C 59BDCCEE3 6B692153 D0A9877C C62A4740 02DF32E5 2139F0A0

5.1.4 摘要算法

本节主要介绍 MD5、SHA1、SM3 算法。

1. MD5

MD5 是 MD4 的改进版，以 512 位分组来处理输入的信息，产生 128 位散列值。

MD5 算法的计算过程主要包括以下步骤。

(1) 预填充

首先,对信息进行填充,使其位长对 512 求余的结果等于 448,即信息的位长 (Bits Length) 被扩展至 $N \times 512 + 448$ 或 $N \times 64 + 56$ 个字节 (N 为一个正整数)。填充的方法为:在信息的后面填充一个 1 和多个 0,直到满足上面的条件。然后,再附加一个 64 位二进制表示的填充前信息长度。经过这两步处理,使消息长度恰好是 512 位的整数倍。

(2) 主循环

主循环的次数为信息中 512 位分组的数目。MD5 中需要 4 个 32 位的链接变量：
A=0x01234567, B=0x89abcdef, C=0xfedcba98, D=0x76543210。

每个主循环有 4 轮 (MD4 只有 3 轮), 每轮循环都很相似。在主循环开始前, 先将 4 个链接变量复制到另外 4 个变量中: A 到 a, B 到 b, C 到 c, D 到 d。第一轮进行 16 次操作。每次操作对 a、b、c 和 d 中的其中 3 个进行一次非线性函数运算, 然后将所得结果加上第 4 个变量、512 位分组数据和一个常数。再将所得结果向右环移一个不定数, 并加上 a、b、c 或 d 中之一。最后用该结果取代 a、b、c 或 d 中之一。

(3) 输出处理

MD5 算法的详细步骤请参见 RFC 1321 (The MD5 Message-Digest Algorithm)。

MD5 ("") = d41d8cd98f00b204e9800998ecf8427e.

MD5 ("a") = 0cc175b9c0f1b6a831c399e269772661。

MD5 ("abc") = 900150983cd24fb0d6963f7d28e17f72.

MD5 ("message digest") = f96b697d7cb7938d525a2f31aaf161d0.

MD5 ("abcdefghijklmnopqrstuvwxyz") = c3fcd3d76192e4007dfb496cca67e13b.

MD5("ABCDEFGHJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789") =
d174ab98d277d9f5a5611c2c9f419d9f.

MD5 ("123456789012345678901234567890123456789012345678901234567890123456")

2. SHA1

SHA1 是以 512 位分组来处理输入信息的，产生 160 位散列值。

SHA1 算法的计算过程与 MD5 类似，主要包括以下步骤。

(1) 预填充

填充方法与 MD5 完全一致。

(2) 主循环

主循环的次数为信息中 512 位分组的数目。SHA1 中需要 5 个 32 位的链接变量：
 $A=0x67452301$, $B=0xefcdab89$, $C=0x98badcfe$, $D=0x10325476$, $E=0xc3d2e1f0$ 。

每个主循环有 4 轮，每轮 20 次操作（MD5 有 4 轮，每轮 16 次操作）。在主循环开始前，先将 5 个链接变量复制到另外 5 个变量中： A 到 a ， B 到 b ， C 到 c ， D 到 d ， E 到 e 。每次操作对 a 、 b 、 c 、 d 、 e 中的 3 个进行一次非线性运算，然后进行与 MD5 中类似的移位运算和加运算。

每个主循环完成以后，将 A 、 B 、 C 、 D 、 E 分别加上 a 、 b 、 c 、 d 、 e ，然后用下一个 512 位分组数据继续进行主循环运算，直至所有分组都完成主循环运算。

(3) 输出处理

将最后一个主循环生成的 A 、 B 、 C 、 D 、 E 这 5 个 32 位值进行级联，生成一个 160 位的摘要值。

SHA1 算法的详细步骤请参见 RFC 3174 (US Secure Hash Algorithm 1 (SHA1))。

验证 SHA1 算法正确性的测试用例如下：

```
SHA1("abc") = A9993E364706816ABA3E25717850C26C9CD0D89D。
SHA1("abcdcbdecdefdefgefghfghighijhi jkijkljklmklmnlmnomnopnopq") =
    84983E441C3BD26EBAAE4AA1F95129E5E54670F1。
SHA1("a")=34AA973CD4C4DAA4F61EEB2BDBAD27316534016F。
SHA1("0123456701234567012345670123456701234567012345670123456701234567")=
    DEA356A2CDDD90C7A7ECEDC5EBB563934F460452。
```

3. SM3

SM3 算法是由中国国家密码管理局于 2012 年 12 月 17 日发布的摘要算法，主要满足电子认证服务系统等应用需求。

针对长度为 l ($l < 2^{64}$) 比特的消息 m ，SM3 算法经过填充和迭代压缩，生成杂凑值，杂凑值长度为 256 位。其中填充过程为：假设消息 m 的长度为 l 位。首先将位“1”添加到消息的末尾，再添加 k 个“0”， k 是满足 $l + 1 + k = 448 \bmod 512$ 的最小的非负整数；然后再添加一个 64 位的比特串，该比特串是长度 l 的二进制表示；填充后的消息 m' 的比特长度为 512 的倍数。

验证 SM3 算法正确性的测试用例如下：

```
SM3("abc") = 66c7f0f4 62eedd9 d1f2d46b dc10e4e2 4167c487 5cf2f7a2 297da02b 8f4ba8e0。
SM3("61626364 61626364 61626364 61626364 61626364 61626364 61626364 61626364")= debe9ff9
    2275b8a1 38604889 c18e5a4d 6fdb70e5 387e5765 293dcba3 9c0c5732。
```


5.2 运算模式（工作模式）

运算模式（Mode of Operation）又称工作模式，是指分组密码算法在处理任意长度消息时，对多个分组进行特殊处理的技术方案。常用的运算模式主要有 ECB、CBC、CFB、OFB 等。

5.2.1 ECB

ECB 是 Electronic Code Book（电子密码本）的缩写。

该模式下，每个分组独立进行加密/解密运算，不同分组之间没有任何关联。于是，在相同密钥的情况下，相同的明文总是产生相同的密文。

该模式在安全性方面的特点是：明文模式不能隐藏、分组加密的输入不是随机的（与明文一样）、一个密钥可以加密多个消息、明文很容易篡改（分组可被删除、再现或互换）等。

该模式在效率方面的特点是：速度与分组算法相同、密文比明文长一个分组（由于填充）、不可能进行预处理、处理过程可并行进行等。

该模式在容错性方面的特点是：一个密文错误会影响整个明文分组、同步错误不可恢复等。

5.2.2 CBC

CBC 是 Cipher Block Chaining（密码分组链接）的缩写。

该模式下，在加密当前分组之前，将上一个分组的加密结果与当前明文分组进行异或后，再进行加密，如此形成一个密文链；解密时类似。

该模式在安全性方面的特点是：明文模式能隐藏（通过与前一个密文分组相异或）、分组加密的输入是随机的（与前一个密文分组相异或）、一个密钥可以加密多个消息、篡改明文稍难（分组可以从消息头或尾处删除，第一个分组位可被更换，并且复制允许控制的改变）等。

该模式在效率方面的特点是：速度与分组算法相同、不考虑 IV 时密文比明文长一个分组、不可能进行预处理、加密不是并行的、解密是并行的且具有随机存取特性等。

该模式在容错性方面的特点是：一个密文错误会影响整个明文分组以及下一个分组的相应位、同步错误不可恢复等。

5.2.3 CFB

CFB 是 Cipher Feed-Back（密码反馈）的缩写。

设对称算法的分组长度为 n 位，初始向量为 IV（ n 位），密钥为 K ，则 x 位 CFB 模式下（ x 在 1 与 n 之间），加密过程如下：

① 设定一个 n 位长的队列，队列初始值为 IV，并将明文消息分成若干个 x 位长的比特块。

② 依次对每个 x 位比特块明文进行以下操作：用密钥 K 对队列进行加密，将该密文中最左端的 x 位与 x 位比特块明文异或，即可获得其 x 位比特块密文；然后将该 x 位比特

块密文放入队列的最右端，并丢弃队列最左端的 x 位。

③ 将所有 x 位比特块密文依次级联，即得到消息密文。

解密过程是加密过程的逆过程。

该模式在安全性方面的特点是：明文模式可以隐藏，分组算法的输入是随机的，用不同的 IV，一个密钥可加密多个消息，篡改明文稍难（分组可以被从消息头或尾处删除，第一个分组位可被更换，并且复制允许控制的改变）等。

该模式在效率方面的特点是：速度与分组算法相同，不考虑 IV 时密文与明文大小相同，在分组出现之前做些预处理是可能的，前面的密文分组可以被加密，加密不是并行的，解密是并行的且有随机性存取特性等。

该模式在容错性方面的特点是：一个密文错误会影响明文的相应位及下一个分组，同步错误是可恢复的，1 位 CFB 能恢复 1 位的添加或丢失等。

5.2.4 OFB

OFB 是 Output Feed-Back（输出反馈）的缩写。

该模式类似于 CFB，主要区别是 OFB 将队列加密后密文中最左端的 x 位放入队列最右端，而 CFB 是将 x 位比特块密文放入队列最右端。

设对称算法的分组长度为 n 位，初始向量为 IV (n 位)，密钥为 K ，则 x 位 OFB 模式下 (x 在 1 与 n 之间) 的加密过程如下：

① 设定一个 n 位长的队列，队列初始值为 IV，并将明文消息分成若干个 x 位长的比特块。

② 依次对每个 x 位比特块明文进行以下操作：用密钥 K 对队列进行加密，将该密文中最左端的 x 位与 x 位比特块明文异或，即可获得其 x 位比特块密文；同时将队列加密后密文中最左端的 x 位放入队列的最右端，并丢弃队列最左端的 x 位。

③ 将所有 x 位比特块密文依次级联，即得到消息密文。

解密过程是加密过程的逆过程。

该模式在安全性方面的特点是：明文模式可以隐藏，分组算法的输入是随机的，用不同的 IV，一个密钥可加密多个消息，明文很容易被控制篡改，任何对密文的改变都会直接影响明文等。

该模式在效率方面的特点是：速度与分组算法相同，不考虑 IV 时密文与明文大小相同，在分组出现之前做些预处理是可能的，OFB 处理过程不是并行的，计数器处理是并行的，等等。

该模式在容错性方面的特点是：一个密文错误仅影响明文的相应位，同步错误不可恢复等。

5.3 扩展机制

5.3.1 MAC 与 HMAC

MAC 是 Message Authentication Code（消息认证码）的缩写。

MAC 机制的工作原理是：消息发送者使用密码算法和密钥对消息进行处理，生成一个

固定大小的小数据块，该数据块称为 MAC 值。消息接收者通过该 MAC 值来验证消息的完整性和来源。

MAC 机制的具体流程如下：

- ① 消息发送者和消息接收者预先协商好密码算法和密钥。
- ② 消息发送者使用已协商的密钥和算法对消息进行处理，生成一个固定大小的 MAC 值。
- ③ 消息发送者将消息和 MAC 值一起发送给消息接收者。
- ④ 消息接收者使用已协商的密钥和算法对消息进行处理，生成新的 MAC 值。
- ⑤ 消息接收者比较新 MAC 值和接收到的 MAC 值，若相同，则确信该消息未被改变且来自消息发送者；若不同，则认为该消息已被改变。

根据使用的密码算法不同，MAC 机制可以分为三类：基于对称算法的 MAC、基于非对称算法的 MAC 和基于 Hash 算法的 MAC。由于已有数字签名机制，一般不使用基于非对称算法的 MAC 机制。目前常用的 MAC 机制有：CBC-DEC-MAC 和 HMAC。前者基于对称算法 DES/3DES，后者基于摘要算法。

1. CBC-DES-MAC

ANSI X9.9 定义 MAC 机制中使用的对称算法为 DES 算法；ANSI X9.19 定义使用双倍长密钥的 3DES 算法计算 MAC 的方法，且它完全与 ANSI X9.9 的单 DES MAC 计算方法兼容。ANSI X9.19 标准使用 MAC 双倍长密钥前半部分对 MAC 数据按 X9.9 计算，然后使用 MAC 密钥后半部分对最后一个分组结果解密计算，再用前半部分加密得到 MAC 值。

ANSI X9.9 和 ANSI X9.19 算法如图 5-5 所示。

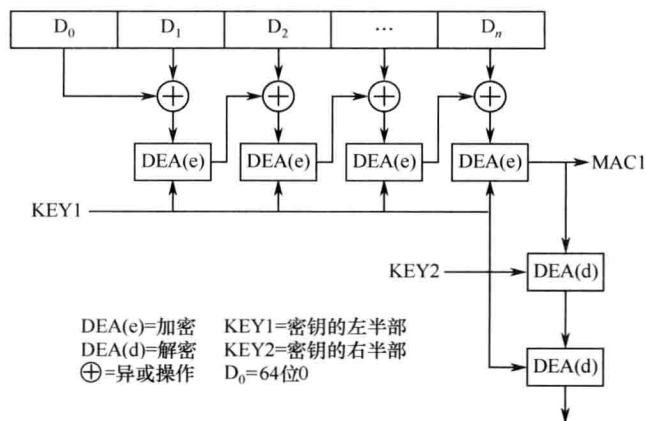


图 5-5 ANSI X9.9 和 ANSI X9.19 算法

其中，MAC1 是 ANSI X9.9 计算的 MAC，MAC2 是 ANSI X9.19 计算的 MAC。对于 MAC 值的长度，ANSI X9.9 等规范要求取最后一个分组密文最左端的 4~8 字节。

不同应用环境下，具体的 MAC 计算方式可能会略有差异。

2. HMAC

HMAC 是 Keyed-Hashing for Message Authentication 的缩写，目前常用的摘要算法是 MD5 或 SHA1。

设 text 表示输入消息, H 表示摘要算法, K 表示密钥, B 表示 H 分组的长度, L 表示 H 输出的长度, ipad=B 个 0x36, opad=B 个 0x5C, 则 HMAC 算法可以表示为:

HMAC 码=H(K XOR opad, H(K XOR ipad, text))。

HMAC 的计算流程描述如下:

- ① 在密钥 K 后面添加 0 来创建一个字长为 B 的字符串。(例如, 如果 K 的字长是 20 字节, B=64 字节, 则 K 后会加入 44 个“0”字节 0x00)。
- ② 将上一步生成的 B 字长字符串与 ipad 做异或运算。
- ③ 将 text 填充至第②步的结果字符串中。
- ④ 用 H 作用于第③步生成的数据流。
- ⑤ 将第①步生成的 B 字长字符串与 opad 做异或运算。
- ⑥ 再将第④步的结果填充进第⑤步的结果中。
- ⑦ 用 H 作用于第⑥步生成的数据流, 输出最终结果, 即为 HMAC 值。

5.3.2 OTP

OTP 是 One-Time Password (一次性口令) 的缩写, 又称动态口令。

OTP 机制的工作原理是: 基于密码算法和密钥, 对同步因子进行处理后, 得到口令; 随时变化的同步因子, 保证了每次产生的口令均不相同, 从而能有效防止口令被攻击或被窃取等安全风险。

OTP 机制的具体流程描述如下:

- ① 用户和服务器预先协商好密码算法和密钥。
- ② 用户获取当前同步因子 (在本地或从服务器获取), 使用已协商的密钥和算法对其进行处理, 生成 OTP 码。
- ③ 用户将 OTP 码发送给服务器。
- ④ 服务器获取当前用户的当前同步因子, 使用已协商的密钥和算法对其进行处理, 生成新的 OTP 码。
- ⑤ 服务器比较新 OTP 码和接收到的 OTP 码, 若相同, 则确信该用户身份合法; 若不同, 则认为该用户身份非法。

OTP 机制中的密码算法可以采用对称算法、非对称算法或摘要算法, 目前比较常用的是对称算法。用户端产生 OTP 码的系统包括: 客户端软件、硬件令牌、手机等。

OTP 机制采用的同步因子主要有以下几类。

1. 时间同步

先保证用户端和服务器端的时间同步, 然后基于用户端或服务器端的本地时间来计算动态口令。该方法对服务器端和用户端的时间精度要求很高; 用户端常用硬件令牌实现, 内嵌高精度的时钟芯片。一般每 30 秒或 60 秒产生一个新的动态口令。

2. 事件同步

先保证用户端和服务器端的事件计数器同步, 然后基于用户端或服务器端的本地事件来计算动态口令。每次计算动态口令后, 用户端和服务器端的事件计数器均要更新。

3. 挑战/应答

先由服务器端产生具有随机性的挑战码，通过网络传递给用户（短信、网站等），用户端基于挑战码计算出应答码，即动态口令；然后服务器端再验证该应答码是否正确。

目前关于 OTP 的国际规范主要有：RFC 1760 (The S/KEY One-Time Password System)。

OTP 机制的优点包括：

- ① 用户再也不能选择脆弱的静态密码。
- ② 用户无需记忆繁多的口令，只需保护好 OTP 产生终端（如令牌、手机等）即可。
- ③ 用户无需担心口令被窃取，因为口令被使用后即失效。
- ④ 能有效阻止网络上针对口令的各种攻击手段。
- ⑤ 便于管理，撤销时仅需要收回令牌或将认证服务器上的对应用户删除即可。
- ⑥ 能有效防止各种口令破解工具的暴力破解。

5.3.3 数字签名

数字签名又称电子签名，数字签名背后的思想是模仿传统手写签名。该思想是能够以某种方式“签署”一份数字文档，该签名具有和物理签名一样的法律效力。与物理世界中的手写签名相对应，数字签名可以理解为数字世界中的电子签名。

在《电子签名法》中，电子签名是指数据电文中以电子形式所含、所附用于识别签名人身份并表明签名人认可其中内容的数据；数据电文是指以电子、光学、磁或者类似手段生成、发送、接收或者储存的信息；民事活动中的合同或者其他文件、单证等文书，当事人可以约定使用或者不使用电子签名、数据电文。

数字签名的功能主要包括：

- ① 接收方能够确认发送方的签名，但不能伪造。
- ② 发送方发出签过名的信息后，不能再否认。
- ③ 接收方对接收到的签名信息也不能否认。
- ④ 一旦发送方和接收方出现争执，仲裁者可有充足的证据进行评判。

数字签名目前只能采用非对称密码算法实现。

数字签名的技术流程描述如下：

- ① 信息发送者使用摘要算法对信息生成信息摘要。
- ② 信息发送者使用自己的私钥对信息摘要进行签名（加密）。
- ③ 信息发送者把信息本身和已签名的信息摘要一起发送出去。
- ④ 信息接收者使用相同的摘要算法对接收的信息本身生成新的信息摘要。
- ⑤ 信息接收者使用信息发送者的公钥对已签名的信息摘要进行验签（解密），获得信息发送者的信息摘要。

⑥ 信息接收者比较这两个信息摘要是否相同，如果相同则确认信息发送者的身份和信息没有被修改过；否则，则表示被修改过。

数字签名的制作和验证过程如图 5-6 所示。

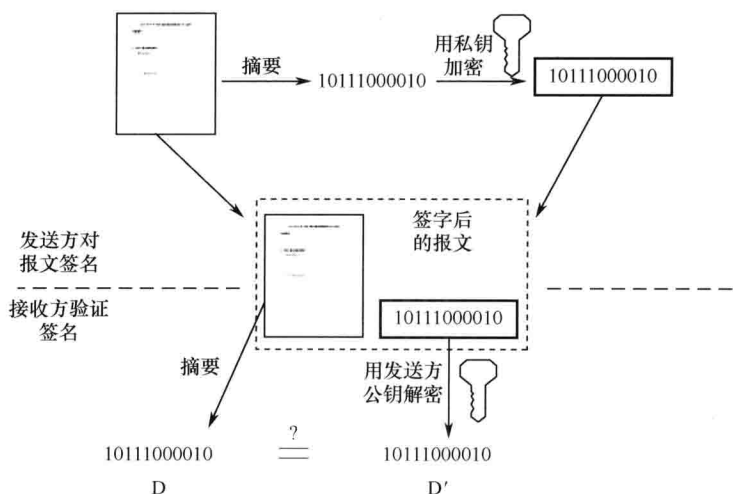


图 5-6 数字签名的制作和验证过程

PKCS #1 规范规定了使用 RSA 算法进行签名时的摘要格式。摘要格式用 ASN.1 描述如下：

```
DigestInfo ::= SEQUENCE {
    digestAlgorithm DigestAlgorithm,
    digest OCTET STRING
}
```

PKCS #7 和 RFC 2315 规范规定了数字签名消息的具体封装格式。数字签名消息封装格式用 ASN.1 描述如下：

```
SignedData ::= SEQUENCE {
    version Version,
    digestAlgorithms DigestAlgorithmIdentifiers,
    contentInfo ContentInfo,
    certificates [0] IMPLICIT ExtendedCertificatesAndCertificates OPTIONAL,
    crls [1] IMPLICIT CertificateRevocationLists OPTIONAL,
    signerInfos SignerInfos }

```

5.3.4 数字信封

对称密码算法的优点是加解密运算非常快，适合处理大批量数据，但其密钥的分发与管理比较复杂。而非对称密码算法的特点是公钥与私钥分离，非常适合密钥的分发与管理；但其运行速度不快，又不适合处理大批量数据。如果将对称密码算法和非对称密码算法的优点结合起来，则既能处理大批量数据，又能简化密钥的分发与管理，于是数字信封机制应运而生。

数字信封并不需要分发和管理对称密钥，而是随机产生对称密钥，采用对称密码算法对大批量数据进行加密，并采用非对称密码算法对该对称密钥进行加密；解密时，先用非对称密码算法解密后获得对称密钥，然后使用对称密码算法解密后获得数据明文。

数字信封的功能类似于普通信封，采用对称密码算法对消息进行加密类似于信纸上的内容，采用非对称密码算法对对称密钥加密类似于信封，信封将信纸包装起来，保证了消息的安全性。

数字信封机制的具体流程如下：

- ① 消息发送方需要预先获得消息接收方的公钥。
- ② 消息发送方随机产生对称密钥，并用该密钥和对称算法对消息进行加密。
- ③ 消息发送方用消息接收方的公钥和非对称算法对上述对称密钥进行加密。
- ④ 消息发送方将消息密文和对称密钥密文一起发送给消息接收方。
- ⑤ 消息接收方收到消息密文和对称密钥密文。
- ⑥ 消息接收方使用自己的私钥和非对称算法对对称密钥密文进行解密后获得对称密钥明文。
- ⑦ 消息接收方使用上述对称密钥和对称算法对消息密文解密后获得消息明文。

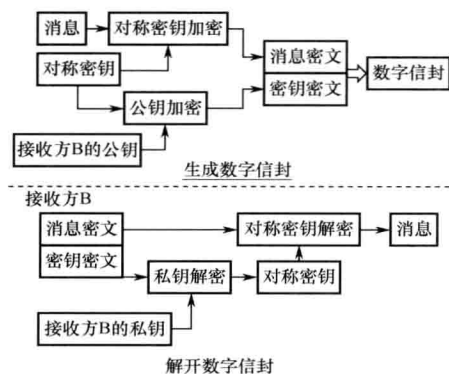


图 5-7 数字信封的生成和解开过程

封装格式。数字信封消息封装格式用 ASN.1 描述如下：

```

EnvelopedData ::= SEQUENCE {
    version Version,
    recipientInfos RecipientInfos,
    encryptedContentInfo EncryptedContentInfo }
RecipientInfos ::= SET OF RecipientInfo
EncryptedContentInfo ::= SEQUENCE {
    contentType ContentType,
    contentEncryptionAlgorithm ContentEncryptionAlgorithmIdentifier,
    encryptedContent [0] IMPLICIT EncryptedContent OPTIONAL }
  
```

5.4 密码应用实践

5.4.1 软件加密与硬件加密

软件加密是指加解密操作全部使用软件模块实现；硬件加密是指加解密操作全部使用专用的硬件设备实现，如加密机、加密卡或 IC 卡等。

软件加密和硬件加密的区别主要体现在以下几个方面。

1. 密钥的安全性

软件加密时，密钥存储在内存或硬盘中，很容易被获取。

而硬件加密时,密钥的生成及保存均在硬件加密设备内部,而且采用了各种防护措施(如开机密钥自毁、密钥成分背对背输入等措施)防止密钥泄露;同时各种密码运算均在加密机内部进行,从而可以对密钥进行根本性的安全保护。

2. 加密过程中的数据安全性

软件加密时很多重要数据或敏感信息(如用来做密码运算的密钥,或顾客的个人密码)都会在某一时刻以明文形式出现在计算机的内存或磁盘中。几乎所有应用业务系统中均采用目前流行的通用型操作系统,安全级别只达到 C2 级,其本身有很多的安全隐患和安全漏洞,而且属于国外产品,其内部留有很多后门已是众所周知,很容易被非法分子或恶意攻击者侵入或控制。所以采用软件加密方式,将使不法分子有机会从不安全的操作系统中对关键数据进行读取、利用、修改或删除,无法保证加密过程中的安全性。

采用硬件加密方式时,所有涉及的重要数据或敏感信息(如用来做密码运算的密钥,或顾客的个人密码)的加解密操作和密码运算均在加密机内部完成,不可能有任何危害客户利益的资料暴露于加密机之外,从而从根本上保证了加密过程中重要数据或敏感信息的安全性。

3. 加密运算速度

软件加密通过在业务主机上运行加密软件来实现加密功能,要占用主机资源,一般来说其运算速度较硬件加密设备要慢很多。

而硬件加密方式是通过独立于主机系统之外的硬件加密设备实现的,硬件加密模块内部本身具有很强大的加解密处理模块,凡涉及加密运算的功能都由硬件加密设备来处理,几乎不占用主机资源,可以显著提高业务系统的处理能力。

4. 运营维护方便性

软件加密模块一般与业务系统运行在相同主机中,模块独立性差,升级维护困难,很难保证业务系统 7×24 小时永不间断的服务宗旨,一旦出现故障,将会给客户的经济利益和社会形象带来巨大影响。

硬件加密模块与各种业务处理系统物理上独立,模块独立性好,升级维护简单,能够从根本上保证业务系统 7×24 小时永不间断的服务宗旨。

5. 密码设备本身的安全性

软件加密自身安全性很难保证。

硬件加密模块能保证自身的安全性。国际上有专门的加密设备安全规范,如 FIPS PUB 140-2 加密模块的安全要求 Level 1~4,国内加密设备在上市前需要通过国家密码管理部门制定的安全测试规范。例如, FIPS 140-2 规定,密码设备必须采用双重控制和知识分离的密钥管理规范;双重控制是指两人或以上的人同时操作才能正确操作密码设备,以防止机密信息被单个人进入密码设备后掌控;知识分离是指由两人或以上的人分别安全保管不同的密钥段,只有同时操作才能重新产生密钥,以防止关键密钥被单个人掌控。FIPS 140-2 还规定,如欲非法获得或修改被输入、保存或处理的敏感信息,必须采用物理方法侵入密码设备;同时要求任何成功的物理侵入将导致密码设备在物理上的损毁,而且无法在不被发现的情况下复原。

5.4.2 网络层加密与应用层加密

按照 TCP/IP 分层模型，网络层加密是指在 IP 层对数据进行加解密处理，而应用层加密是指在应用层对数据进行加解密处理。

网络层加密主要采用 IP 加密技术，对 IP 层的所有数据进行加密，与应用系统无关。其优点是对应用系统透明、实施简单、能防止外部攻击；缺点是与应用系统脱离，无法做到选择性的加密保护，无法做到防止内部攻击，也无法保证敏感数据的全程安全性。

应用层加密的优点是能够做到选择性的加密保护，能保证敏感数据的全程安全性，能够同时防止外部攻击和内部攻击；其缺点是与应用系统结合紧密、开发工作量大、实施难度大。

网络层加密和应用层加密在保护数据全程安全性方面的比较见表 5-4。

表 5-4 网络层加密和应用层加密的比较

比较项		网络层加密	应用层加密
数据存储安全		不能	能
数据传输安全	主干网	能	能
	局域网	不能	能
数据处理安全		不能	能
数据输入安全		不能	能

5.4.3 密钥管理的基本原则

密钥管理主要遵循以下基本原则。

1. 保证密钥自身的机密性（Keys Are Secret）

除非对称公钥外，用于加解密的密钥和用于生成密钥的敏感资料必须保持机密，不允许任何人知道任何密钥。

2. 限制密钥的存在形式（Keys Have Permissible Forms）

除非对称公钥外，用于加解密的密钥只允许以下几种存在形式：

- ① 密钥明码（Clear Text）只能存在于硬件加密设备中。
- ② 在硬件加密设备外，密钥必须加密存储。
- ③ 在硬件加密设备外的密钥成分（用于合成密钥），必须保证双重控制、多人掌握。

3. 密钥分发（Key Deployment）

用于加解密的密钥，在进行分发时必须保证在切实可信、最少数目的地点进行。

4. 密钥隔离（Key Separation）

用于加解密的密钥，在产生和使用时必须用于其最初设计的目的。

5. 密钥同步（Key Synchronization）

必须提供机制来保证和验证已分发密钥的正确性，且该密钥的使用不会影响其他密钥的安全性。

6. 活动日志 (Event Journal)

密钥管理的所有活动必须记录到活动日志中,且该日志库必须保证自身管理的安全性。

5.4.4 密码设备的自身安全性

FIPS 140 标准对密码模块自身安全性做出了具体的技术规定。该标准由美国国家标准和技术委员会 (NIST) 发布,专门针对密码模块的安全需求 (Security Requirements for Cryptographic Modules)。目前该标准的最新版本 (即 FIPS 140-2) 发表于 2002 年 12 月 3 日,其提供了密码模块评测、验证和最终认证的基础。NIST 正在进行该标准新版本的审核,未来将发布 FIPS PUB 140-3。FIPS 140 标准已被 ISO 标准所采用。

1. 密码模块自身安全性的基本要求

- ① 采用并正确执行经批准的保护敏感信息的安全功能。
- ② 保护一个加密模块,防止未经授权的操作或使用。
- ③ 防止加密模块与算法被未经授权或不被发现地篡改。
- ④ 提供加密模块运行状态指示。
- ⑤ 当加密模块在一个已认可的操作模式下运行时,保证加密模块的正常运行。
- ⑥ 能检测到加密模块在运行中产生的错误,并防止这些错误导致对关键数据的损害。

2. 安全等级划分

鉴于所要保护敏感数据的价值不同以及应用环境的多样性,密码模块在安全方面的要求也有很大差异。为了满足不同等级敏感信息和不同应用环境的安全需求,FIPS 140 将密码模块的安全级别分为 4 级,由低到高依次为 Level 1、Level 2、Level 3、Level 4。

(1) Level 1

Level 1 提供最低级别的安全性。在 Level 1 中,基本没有产品级元器件之外的安全控制功能。Level 1 允许一个密码模块的软件和固件成分,在一般用途的计算系统上运行 (操作系统可能未经安全验证)。Level 1 密码设备适合一些低安全要求的应用,此时诸如物理安全、网络安全、管理程序安全限制等安全控制手段很有限或根本不存在。

(2) Level 2

Level 2 加强了 Level 1 密码模块在物理结构上的安全要求,通过提供能够识别的入侵证据,以防止明显的破坏。这些可以识别的入侵证据包括:可识别入侵的涂层或密封、在可移动的盖子或门上加锁等,并且只有破坏涂层、密封等物理防护手段,才能从物理上进入密码模块组件内部,从而获取密钥明文等关键参数。

Level 2 要求至少基于角色的验证机制。通过该验证机制,密码模块可以验证操作员的角色和权限。

(3) Level 3

Level 3 除了要求 Level 2 的物理结构外,还要求防止对密码模块内部所控制的关键安全参数过程的攻击。对企图通过物理登陆、使用或篡改密码模块的攻击活动,密码模块能够检测并及时做出反应,当可移动的门或盖子被非法打开时能够清除密码模块中的明文关键安全参数。

Level 3 要求基于身份识别的验证机制,通过此验证机制,密码模块可以验证操作员的身份、角色和权限。

Level 3 要求明文关键参数的输入和输出时，所使用的端口必须与其他端口物理上分开，或者使用其他可信的方式进行逻辑分离。关键安全参数可以以密文的方式从密码模块输入或输出。

(4) Level 4

Level 4 是最高级别的安全要求。在此级别上，物理安全机制需要一个完整的保护封装。与 Level 3 相比，Level 4 对物理入侵的要求更为严格，对贯穿封装的任意方向攻击进行高灵敏度的检测，并能立即清除所有在密码模块中的明文关键参数。

Level 4 的密码模块可以在物理上无防护的环境中操作使用。这要求密码模块既能够适应正常的操作范围环境需求（如温度、电压），又能检测具有风险的环境波动并清除关键安全参数。

3. 安全要求要点

密码模块安全要求主要包括：设计与运行，组件规格，连接端口与接口，角色、服务与验证，有限状态模式，物理安全性，操作环境，密钥管理，电磁干扰/电磁兼容性，自检与设计保证。对于不同安全级别，安全要求有所不同，具体比较参见表 5-5。

表 5-5 密码模块安全要求

	Level 1	Level 2	Level 3	Level 4
组件规格	密码模块规格、安全范围边界、认可的加密算法、认可的运行模式、密码模块说明、组件安全策略描述			
连接接口与端口	要求提供所有接口的规格与输入、输出数据的路径		对于未保护的关键安全参数端口，应当在逻辑上与其他数据连接端口分开	
角色、服务与验证	要求逻辑上分开的可选角色与服务	基于角色的操作员验证	基于身份识别的操作员验证	
有限状态模式	有限状态模式的规格，要求的规定与可选的规定，规定转换图与规定转换条件			
物理安全性	生产合格设备	锁或入侵证据	对外壳与门的入侵检测与反应	入侵检测与反应的封装，EFP 或 EFT
操作环境	单独操作员，可执行代码，认可的集成技术	在 EAL2 中引用的外壳防护评估，自由选择访问控制机制和检测	在 EAL3 中引用的外壳防护与可信路径评估，加安全策略模式	在 EAL4 中引用的外壳防护与可信路径评估
密钥管理	人工方式建立的密钥可以明文方式输入和输出		人工方式建立的密钥应当以加密或知识分离的方式输入和输出	
EMI/EMC	47CFR FCC15, B 分册, A 类（商用）		47CFR FCC15, B 分册, B 类（家用）	
设计保证	配置管理,安全安装与设置,设计与策略一致,指导文件	配置管理, 安全配置、功能规格	高级语言的实现	正式型号，详细说明，预处理和后处理

5.5 密码算法 ASN.1 描述

5.5.1 密码算法格式

密码算法格式用 ASN.1 定义如下：

```
AlgorithmIdentifier ::= SEQUENCE {
    Algorithm          OBJECT IDENTIFIER,
```

Parameters ANY DEFINED BY algorithm OPTIONAL }

其中, Algorithm 为算法 OID, Parameters 为算法参数。

5.5.2 密码算法 OID

常用算法的 OID 表 5-6。

表 5-6 常用算法 OID

/	算 法	OID	/	算 法	OID
1	md2	1.2.840.113549.2.2	13	md5WithRSAEncryption	1.2.840.113549.1.1.4
2	md4	1.2.840.113549.2.4	14	sha1WithRSAEncryption	1.2.840.113549.1.1.5
3	md5	1.2.840.113549.2.5	15	sha256WithRSAEncryption	1.2.840.113549.1.1.11
4	sha1	1.3.14.3.2.26	16	sha384WithRSAEncryption	1.2.840.113549.1.1.12
5	sha256	2.16.840.1.101.3.4.2.1	17	sha512WithRSAEncryption	1.2.840.113549.1.1.13
6	sha384	2.16.840.1.101.3.4.2.2	18	sm3WithSM2Encryption	1.2.156.10197.1.501
7	sha512	2.16.840.1.101.3.4.2.3	19	pbeWithMD2AndDES-CBC	1.2.840.113549.1.5.1
8	sm3		20	pbeWithMD2AndRC2-CBC	1.2.840.113549.1.5.4
9	rsaEncryption	1.2.840.113549.1.1.1	21	pbeWithMD5AndDES-CBC	1.2.840.113549.1.5.3
10	sm2	1.2.156.10197.1.301	22	pbeWithMD5AndRC2-CBC	1.2.840.113549.1.5.6
11	md2WithRSAEncryption	1.2.840.113549.1.1.2	23	pbeWithSHA1AndDES-CBC	1.2.840.113549.1.5.10
12	md4WithRSAEncryption	1.2.840.113549.1.1.3	24	pbeWithSHA1AndRC2-CBC	1.2.840.113549.1.5.11

5.6 密码消息 ASN.1 描述

PKCS #7 规范和《SM2 密码算法加密签名消息语法规则》中规定了各种密码消息的具体格式, 这些消息可用于不同实体间的数据交换。

5.6.1 通用内容消息 ContentInfo

通用内容消息格式用 ASN.1 描述如下:

```
ContentInfo ::= SEQUENCE {
    contentType ContentType,
    content [0] EXPLICIT ANY DEFINED BY contentType OPTIONAL }
ContentType ::= OBJECT IDENTIFIER
```

其中, contentType 表示内容类型或消息类型, 包括 data、signedData、envelopedData、signedAndEnvelopedData、digestData、encryptedData、keyAgreementInfo 等。content 表示内容值。

5.6.2 明文数据消息 Data

明文数据消息格式 ASN.1 描述如下:

```
Data ::= OCTET STRING
```

5.6.3 数字签名消息 SignedData

1. SignedData

数字签名消息格式用 ASN.1 描述如下：

```
SignedData ::= SEQUENCE {
    version Version,
    digestAlgorithms DigestAlgorithmIdentifiers,
    contentInfo ContentInfo,
    certificates [0] IMPLICIT ExtendedCertificatesAndCertificates OPTIONAL,
    crls [1] IMPLICIT CertificateRevocationLists OPTIONAL,
    signerInfos SignerInfos }
DigestAlgorithmIdentifiers ::= SET OF DigestAlgorithmIdentifier
DigestAlgorithmIdentifier ::= AlgorithmIdentifier
SignerInfos ::= SET OF SignerInfo
ExtendedCertificatesAndCertificates ::= SET OF ExtendedCertificatesAndCertificate
ExtendedCertificatesAndCertificate ::= CHOICE {
    certificate Certificate, -- X.509
    extendedCertificate [0] IMPLICIT ExtendedCertificate }
```

使用 SM2 算法时，SignedData 用 ASN.1 描述如下：

```
SignedData ::= SEQUENCE {
    version Version,
    digestAlgorithms DigestAlgorithmIdentifiers,
    contentInfo SM2Signature,
    certificates [0] IMPLICIT ExtendedCertificatesAndCertificates OPTIONAL,
    crls [1] IMPLICIT CertificateRevocationLists OPTIONAL,
    signerInfos SignerInfos }
```

其中，version 表示消息格式版本，缺省值为 1。digestAlgorithms 是摘要算法标识集合，可以包含零或多个摘要算法标识。contentInfo 表示待签名数据。certificates 包含签名者认证路径中的相关证书。crls 是 CRL 集合。signerInfos 是签名者信息集合，至少包含一个签名者。

2. SignerInfo

签名者信息格式用 ASN.1 描述如下：

```
SignerInfo ::= SEQUENCE {
    version Version,
    issuerAndSerialNumber IssuerAndSerialNumber,
    digestAlgorithm DigestAlgorithmIdentifier,
    authenticatedAttributes [0] IMPLICIT Attributes OPTIONAL,
    digestEncryptionAlgorithm DigestEncryptionAlgorithmIdentifier,
    encryptedDigest EncryptedDigest,
    unauthenticatedAttributes [1] IMPLICIT Attributes OPTIONAL }
IssuerAndSerialNumber ::= SEQUENCE {
```

```

    issuer Name,
    serialNumber CertificateSerialNumber }
DigestEncryptionAlgorithmIdentifier ::= AlgorithmIdentifier
EncryptedDigest ::= OCTET STRING

```

其中, `version` 表示消息格式版本, 缺省值为 1。`issuerAndSerialNumber` 用于唯一确定签名者证书, 由其证书签发者 DN 和证书序列号组成。`digestAlgorithm` 表示摘要算法标识, 应属于 `SignedData` 中的 `digestAlgorithms` 摘要算法集合。`authenticatedAttributes` 表示签名者用于签名的属性集合。`digestEncryptionAlgorithm` 表示用签名者私钥对摘要加密或签名的公钥算法进行标识。`EncryptedDigest` 表示用签名者私钥对摘要进行加密或签名后的值; 当使用 SM2 算法时, 为 SM2Signature 类型, 编码格式为 r||s。`unauthenticatedAttributes` 表示签名者未用于签名的属性集合。

5.6.4 数字信封消息 EnvelopedData

1. EnvelopedData

数字信封消息格式用 ASN.1 描述如下:

```

EnvelopedData ::= SEQUENCE {
    version Version,
    recipientInfos RecipientInfos,
    encryptedContentInfo EncryptedContentInfo }
RecipientInfos ::= SET OF RecipientInfo

```

其中, `version` 表示消息格式版本, 缺省值为 0。`recipientInfos` 是接收者信息集合, 至少包含一个接收者。`encryptedContentInfo` 表示已加密的内容信息。

`encryptedContentInfo` 格式用 ASN.1 描述如下:

```

encryptedContentInfo ::= SEQUENCE {
    contentType ContentType,
    contentEncryptionAlgorithm ContentEncryptionAlgorithmIdentifier,
    encryptedContent [0] IMPLICIT EncryptedContent OPTIONAL }
ContentEncryptionAlgorithmIdentifier ::= AlgorithmIdentifier
EncryptedContent ::= OCTET STRING

```

其中, `contentType` 表示内容类型。`contentEncryptionAlgorithm` 表示密码算法标识, 用于加密内容; 针对所有接收者, 采用相同的内容加密算法。`encryptedContent` 表示加密后的内容。

使用 SM2 算法时, `encryptedContentInfo` 用 ASN.1 描述如下:

```

encryptedContentInfo ::= SEQUENCE {
    contentType ContentType,
    contentEncryptionAlgorithm ContentEncryptionAlgorithmIdentifier,
    encryptedContent [0] IMPLICIT EncryptedContent OPTIONAL,
    sharedInfo [1] IMPLICIT OCTET STRING OPTIONAL,
    sharedInfo [2] IMPLICIT OCTET STRING OPTIONAL
}

```

其中, `sharedInfo` 表示发送者和接收者之间协商好的共享信息。

2. RecipientInfo

接收者信息格式用 ASN.1 描述如下：

```
RecipientInfo ::= SEQUENCE {
    version Version,
    issuerAndSerialNumber IssuerAndSerialNumber,
    keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,
    encryptedKey EncryptedKey }
KeyEncryptionAlgorithmIdentifier ::= AlgorithmIdentifier
EncryptedKey ::= OCTET STRING
```

其中，version 表示消息格式版本，缺省值为 0。issuerAndSerialNumber 用于唯一确定接收者证书，由其证书签发者 DN 和证书序列号组成。keyEncryptionAlgorithm 表示密码算法标识，用于加密密钥；被加密的密钥用于加密内容，加密后的内容保存在 encryptedContentInfo→encryptedContent 中。EncryptedKey 表示被加密的密钥。

5.6.5 数字签名及信封消息 SignedAndEnvelopedData

数字签名及信封消息格式用 ASN.1 描述如下：

```
SignedAndEnvelopedData ::= SEQUENCE {
    version Version,
    recipientInfos RecipientInfos,
    digestAlgorithms DigestAlgorithmIdentifiers,
    encryptedContentInfo EncryptedContentInfo,
    certificates [0] IMPLICIT ExtendedCertificatesAndCertificates OPTIONAL,
    crls [1] IMPLICIT CertificateRevocationLists OPTIONAL,
    signerInfos SignerInfos }
```

其中，version 表示消息格式版本，缺省值为 1。recipientInfos 是接收者信息集合，至少包含一个接收者。digestAlgorithms 是摘要算法标识集合，可以包含零或多个摘要算法标识。encryptedContentInfo 表示已加密的内容信息。certificates 包含签名者认证路径中的相关证书。crls 是 CRL 集合。signerInfos 是签名者信息集合，至少包含一个签名者。

5.6.6 摘要消息 DigestedData

摘要消息格式用 ASN.1 描述如下：

```
DigestedData ::= SEQUENCE {
    version Version,
    digestAlgorithm DigestAlgorithmIdentifier,
    contentInfo ContentInfo,
    digest Digest }
Digest ::= OCTET STRING
```

其中, version 表示消息格式版本, 缺省值为 0。digestAlgorithm 是摘要算法标识。contentInfo 包含待摘要内容。digest 表示摘要值。

5.6.7 加密数据消息 EncryptedData

加密数据消息格式用 ASN.1 描述如下:

```
EncryptedData ::= SEQUENCE {
    version Version,
    encryptedContentInfo EncryptedContentInfo }
```

其中, version 表示消息格式版本, 缺省值为 0。encryptedContentInfo 表示已加密的内容信息。

5.6.8 密钥协商消息 KeyAgreementInfo

密钥协商消息用于两个用户之间为协商共享密钥进行公共参数交换, 仅用于 SM2 算法, 具体格式用 ASN.1 描述如下:

```
KeyAgreementInfo ::= SEQUENCE {
    version      Version (1),
    tempPublicKey SM2PublicKey,
    userCertificate Certificate,
    userID       OCTET STRING }
```

其中, version 表示消息格式版本, 缺省值为 1。tempPublickeyR 表示临时公钥。userCertificate 表示用户证书。userID 表示用户标识。

5.6.9 密码消息类型 OID

密码消息类型的 OID 见表 5-7。

表 5-7 密码消息类型 OID

/	消息类型	OID
1	data	1.2.840.113549.1.7.1 1.2.156.10197.6.1.4.2.1 (仅 SM2 算法)
2	signedData	1.2.840.113549.1.7.2 1.2.156.10197.6.1.4.2.2 (仅 SM2 算法)
3	envelopedData	1.2.840.113549.1.7.3 1.2.156.10197.6.1.4.2.3 (仅 SM2 算法)
4	signedAndEnvelopedData	1.2.840.113549.1.7.4 1.2.156.10197.6.1.4.2.4 (仅 SM2 算法)
5	digestedData	1.2.840.113549.1.7.5
6	encryptedData	1.2.840.113549.1.7.6 1.2.156.10197.6.1.4.2.5 (仅 SM2 算法)
7	keyAgreemengInfo	1.2.156.10197.6.1.4.2.6 (仅 SM2 算法)

注: pkcs-7 OBJECT IDENTIFIER ::= { iso(1) member-body(2) US(840) rsadsi(113549) pkcs(1) 7 }

5.7 Base64 编码

Base64 是一种很常用的编码方式,可以将任意二进制字节编码成 64 个可打印的 ASCII 码字符。Base64 编码的基本原理如下。

1. 字符索引

64 个字符包括大小写英文字母各 26 个、10 个数字、加号和左斜杠。等号用于末尾填充。64 个字符所对应的索引如表 5-8 所示。

表 5-8 Base64 字符索引表

索引	字符	索引	字符	索引	字符	索引	字符
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

2. 编码过程

将二进制字节流按 3 个字节一组进行分组,然后分别对每个 3 字节组进行编码。当二进制字节流长度不是 3 的倍数时,最后一个 3 字节组可能不足 3 个字节,编码时需要进行填充处理。每个 3 字节组可编码成 4 个字符。

将 3 字节组中的每个字节按最高位到最低位顺序排列后组成 24 位,然后按 6 位一组分成 4 组,每组取值范围为 0~63,将每组取值作为索引按照表 5-7 获得对应的字符后,最终形成 4 个字符。

如果最后的 3 字节组只有 1 个字节,则将形成 8 位,右边补足 4 位 0 比特后,可形成 2 个 6 位组,转换成 2 个字符,然后填充 2 个等号,凑成 4 个字符。如果最后的 3 字节组只有 2 个字节,将形成 16 位,右边补足 2 位 0 比特后,可形成 3 个 6 位组,转换成 3 个字符,然后填充 1 个等号,凑成 4 个字符。

Base64 编码后的字符之间允许插入回车换行符,但每行不允许超过 76 个字符。

3. 解码过程

将字符流中回车换行符删除后,按4个字符一组进行分组,然后分别对每个4字符组进行解码。若删除回车换行符后的字符流长度不是4的倍数,则该字符流有误。每个4字符组可解码成3个字节。若最后一个4字符组末尾是1或2个等号,将解码成2或1个字节。

将4字符组中每个字符按表5-8获得对应的索引,该索引取值范围为0~63,将索引转换成6位比特后,按最高位到最低位顺序排列后组成24位比特;然后按8位一组分成3组,每组8位转换成1个字节,最终形成3个字节。

如果最后的4字符组末尾是1个等号,则前3个字符的索引将组合成18位,取前16位转换成2个字节。如果最后的4字符组末尾是2个等号,则前2个字符的索引将组合成12位,取前8位比特转换成1个字符。

第6章 LDAP技术

6.1 目录服务与LDAP概述

6.1.1 目录服务简介

在介绍目录服务之前，先回顾一下目录的概念。在现实生活中，我们会接触哪些与目录相关的东西呢？最常见到的目录是电话簿，电话簿中按照字母表次序列出人名以及他们的电话号码和地址，如果是企业，则电话簿中可能会有这些企业的传真号码或营业时间等信息。

另一个目录例子是图书馆的图书检索卡，在图书馆，每本图书都对应一张检索卡，检索卡上记录了图书的名字、作者姓名、出版日期、页数、价格、存放位置等内容。这些检索卡按照一定顺序进行排列，通过检索卡指示的位置，借阅人就可以找到图书。

还有一个常见的使用目录的例子是通过字典检索表查找文字，可以使用拼音进行检索，每个发音对应一个条目，条目列出了对应的文字的页码；也可以使用部首进行检索，依据部首笔画，找出对应的文字及页码。

虽然这些例子各不相同，但可以用一种方式进行统一，即每个目录可以看作一个对象，这个对象包含一个或多个属性。如电话号码簿例子中，电话号码和地址是这个对象的属性；在图书检索卡中，图书的名字、作者姓名、出版日期等都是属性；在字典例子中，页码是属性。

目录的一种更形式化的定义是：一个目录是指一组名字和值的映射，它允许根据一个给出的名字来查找对应的值。与词典相似，每一个词也许会有多个词义，在一个目录中，一个名字也许会与多个不同的信息相关联。类似地，就像一个词会有多个不同的发音和多个不同的词义，目录中的一个名字可能会有多个不同类型的值。

有了目录的定义，目录服务的定义就比较清楚了：目录服务由目录数据库和一套访问协议组成，是一个存储、组织和提供信息访问服务的软件系统，目录服务本质上是一种基于客户/服务器模型的信息查询、管理服务。目录数据库存储目录数据，它以树状的层次结构描述目录数据信息。

目录服务有如下特点：

- ① 为读操作进行高度优化。
- ② 可采用分布模式存储信息。
- ③ 可扩展其存储的信息类型。
- ④ 具备高级查询能力。
- ⑤ 支持信息在不同目录服务之间复制。

与关系数据库相比，目录服务更擅长查询。目录数据库中的数据读取和查询效率非常高，比关系型数据库能够快一个数量级，但数据写入效率相对较低，适用于数据不需要经常改动，但需要频繁读出的情况。

目录服务的一个简单实现是名字服务，名字服务将一个网络资源的名字与它的网络地

址进行了映射。采用名字服务这种类型的目录，用户不必记住某个网络资源的物理地址，只需要提供这个网络资源的名字就可以找到它。在网络上的每一个资源都被目录服务当作一个对象，关于某个网络资源的信息被作为这个对象的属性存储起来。存储到对象之内的信息可以进行访问控制以增强安全性，这样只有授权的用户才能访问到这些信息。

目录服务遵循 LDAP 和 X.500 协议。一些厂商提供了自己的目录服务产品，如微软公司的 Active Directory、Novell 公司的 eDirectory、IBM 的 Tivoli Directory、开源的 OpenLDAP 等。

现在目录服务用于管理各种大量的信息，其中包括 QoS、带宽管理策略、配置文件、电子商务信息及其他信息，在用户的身份验证、防火墙过滤及 VPN 访问方面也起着重要的安全作用。

目录服务在电子商务和企业对企业之间的关系中扮演着重要角色。目录可保存有关公司网络外部人员的重要信息，用于鉴别这些人员并定义他们对于网络资源的访问权。

6.1.2 X.500 协议简介

X.500 是国际电信联盟 (ITU-T) 定义的目录标准，它主要充当商业产品的一种模型。设计 X.500 的目的是使用 OSI 协议族，但是 TCP/IP 却成为了实际的网络协议，因此，如今大多数目录服务都以 X.500 为模型并设计用于在 TCP/IP 上运行。

X.500 是一个协议族，由一系列的概念和协议组成，包括：

- ① X.501：模型定义，定义目录服务的基本模型和概念。
- ② X.509：认证框架，定义如何处理目录服务中的客户和服务器认证。
- ③ X.511：抽象服务定义，定义 X.500 提供的功能性服务。
- ④ X.518：分布式操作过程定义，定义如何跨平台处理目录服务。

⑤ X.519：协议规范，定义了 X.500 协议，包括 DAP (Directory Access Protocol, 目录访问协议)、DSP (Directory System Protocol, 目录系统协议)、DOP (Directory Operator Protocol, 目录操作绑定协议)、DISP (Directory Information Shadowing Protocol, 目录信息阴影协议)。

- ⑥ X.520：定义属性类型要求。
- ⑦ X.521：定义对象类型。
- ⑧ X.525：定义如何在目录服务器间复制内容。

X.500 标准中定义了很多内容，包括：

- ① 定义信息模型，确定目录中信息的格式和字符集，如何在项中表示目录信息（定义对象类、属性等模式）。
- ② 定义命名空间，确定对信息进行的组织和引用，如何组织和命名项、目录信息树 (DIT, Directory Information Tree) 和层次命名模型。
- ③ 定义功能模型，确定可以在信息上执行的操作。
- ④ 定义认证框架，保证目录中信息的安全，以及如何实现目录中信息的授权保护（访问控制模型）。
- ⑤ 定义分布操作模型，确定数据如何分布和如何对分布数据执行操作，如何将全局目录树划分为管理域，以便管理。

⑥ 定义客户端与服务器之间通信的各种协议。

X.500 主要具备以下特征。

① 分散维护：运行 X.500 的每个站点只负责其本地目录部分，所以可以立即进行更新和维护操作。

② 强大的搜索性能：X.500 提供强大的搜索功能，支持用户建立的任意复杂查询。

③ 单一全局命名空间：类似于 DNS，X.500 为用户提供单一的相同命名空间。与 DNS 相比，X.500 的命名空间更灵活且易于扩展。

④ 结构化信息结构：X.500 目录中定义了信息结构，允许本地扩展。

⑤ 基于标准的目录服务：由于 X.500 可以被用于建立一个基于标准的目录，那么在某种意义上，请求应用目录信息（电子邮件、资源自动分配器、特定目录工具）的应用程序就能访问重要且有价值的信息。

X.500 虽然是一个完整的目录服务协议，但在实际应用的过程中却存在着不少障碍。由于目录访问协议 DAP 这种应用层协议是严格遵照复杂的 ISO 七层协议模型制定的，对相关层协议环境要求过多，主要运行在 UNIX 机器上，在许多小系统上，如 PC 和 Macintosh 上无法使用，因此没有多少人按照 DAP 开发应用程序，TCP/IP 协议体系的普及更使得这种协议越来越不适应需要。

由于 X.500 的实施太过复杂而受到批评。为解决这个问题，密歇根州立大学推出了一种较为简单的基于 TCP/IP 的 DAP 新版本，即轻量级目录访问协议（LDAP，Lightweight Directory Access Protocol），主要用于 Internet。LDAP 与 DAP 具有很多类似的基本功能，另外它还能用来查询私有目录和开放 X.500 目录上的数据。在过去的几年里，大多数主要的电子邮件和目录服务软件供应商都对 LDAP 表现出了极大的兴趣，LDAP 已迅速发展成为 Internet 上事实的目录协议标准。

6.1.3 LDAP 协议简介

LDAP 的目的很明确，就是要简化 X.500 目录的复杂度以降低开发成本，同时适应 Internet 的需要。LDAP 已经成为目录服务的标准，它比 X.500 DAP 协议更为简单实用，而且可以根据需要定制，因而实际应用也更为广泛。

与 X.500 协议相比，LDAP 在 4 个方面对 X.500 的 DAP 进行了简化：

① 功能方面。LDAP 提供 DAP 大多数功能的较低开销实现，去掉了 DAP 的冗余操作和很少使用的功能，简化了客户端和服务端端的实现。

② 数据表示方面。在 LDAP 中，大多数数据元素使用简单字符串表示，简化了实现和提高了效率。当然，为了提高效率，字符串包装在二进制编码的消息中。

③ 编码方面。使用 X.500 的编码规则子集对 LDAP 消息编码，从而能够简化实现。

④ 传输协议。LDAP 使用 TCP 传输协议，而不是 OSI 多层网络协议栈，实现得到简化，性能得到提升，完全去除了对 OSI 的依赖，使 LDAP 目录的部署更加简单。

X.500 采用公钥基础结构（PKI）作为主要的认证方式，而 LDAP 最初并不考虑安全问题，目前已增加了安全机制。为保证数据访问安全，可使用 LDAP 的 ACL（Access Control List，访问控制列表）来控制对数据读和写的权限。

LDAP 目前有第 2 版 LDAP v2 和第 3 版 LDAP v3 两个版本，基于 LDAP v3 的服务器可以让用户使用支持 LDAP 功能的 Web 浏览器，进行有关电子邮件用户的查询，可以查询的用户属性包括姓名、电话号码、电子邮件地址和地址信息等；系统管理员可以通过 LDAP 客户程序远程进行目录管理操作，如添加、删除和修改用户账户信息等；可以请求服务器执行扩展操作。

6.1.4 LDAP 模型简介

LDAP 模型是从 X.500 协议中继承而来的，是 LDAP 的一个组成部分，用于指导客户如何使用目录服务。LDAP 定义了 4 个模型，包括信息模型（Information Model）、命名模型（Naming Model）、功能模型（Functional Model）、安全模型（Security Model）。

1. 信息模型

LDAP 信息模型用于描述 LDAP 中信息的表达方式，包含 3 部分：条目（Entries）、属性（Attributes）、值（Values）。LDAP 使用专有逻辑格式存储信息，这种模型既不是关系的，也不是完全面向对象的。简单概括为：

- ① LDAP 中信息逻辑上表示为条目。
- ② 条目包含一到多个对象类。
- ③ 每个对象类由多个属性组成。
- ④ 每个属性包含一到多个同一类型的数值。
- ⑤ 对象类和属性的类型定义构成了 schema。

条目是目录中最基本的信息单元，可以理解为目录树中的一个节点。LDAP 客户和服务使用条目共享信息，条目是 LDAP 服务器的基本元素。执行搜索时服务器返回一组匹配条目，但修改时一次只能影响服务器中的一个条目。条目可以被任何支持 LDAP 的客户端创建，或通过使用服务器工具导入，也可由应用程序基于非 LDAP 数据或用户输入信息创建。

在目录中添加一个条目时，该条目必须包含一个或多个对象类（objectClass），每一个对象类规定了该条目中的必选属性和可选属性。图 6-1 展示了一个典型目录的一部分，它反映了现实世界中一个组织的管理对象。

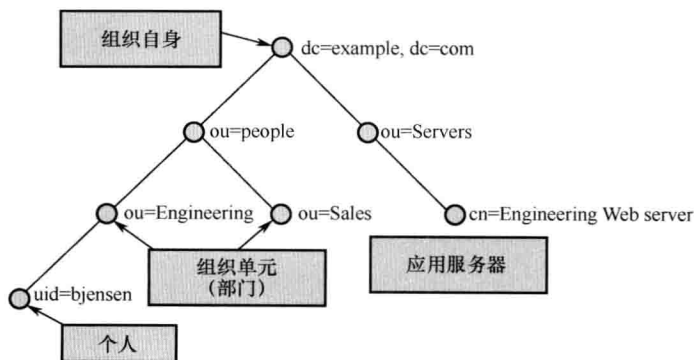


图 6-1 组织结构目录树

每一个条目都有一个 DN (distinguished name, 辨别名), 用于唯一标识条目在目录中的位置, 本书将在命名模型中详细介绍 DN。

每个条目都是由多个属性组成的, 每一个属性描述了对象的一个约束。每个属性具有一个类型和一个或多个值。该类型描述了包含在属性中的信息类型, 其值中包含实际数据。例如, 表 6-1 描述了一个人员条目, 属性包括全称、名、姓、电话号码、电子邮件地址。

表 6-1 人员属性

属性类型	属性值
cn	张三
telephoneNumber	58046690
	62300098
mail	zs@163.com

属性有与之关联的语法和匹配规则, 属性语法指定可以在属性中存放的数据格式, 如 INTEGER 语法允许值中只包含数字, 不能包含非数字字符。

匹配规则作用有二: 第一, 比较值是否相等; 第二, 对值进行排序。

2. 命名模型

LDAP 命名模型定义了如何在目录系统中组织数据以及如何从目录系统中查找数据。命名模型的灵活性可以使你很方便地以想要的方式组织数据。例如, 可以把组织中所有人员放在一个容器下, 所有组放在一个容器下, 或按组织结构的地理分布组织目录结构。

LDAP 命名模型指定将条目按类似倒立的树结构进行规划, 非常类似于 UNIX 系统的文件系统, 如图 6-2 和图 6-3 所示。

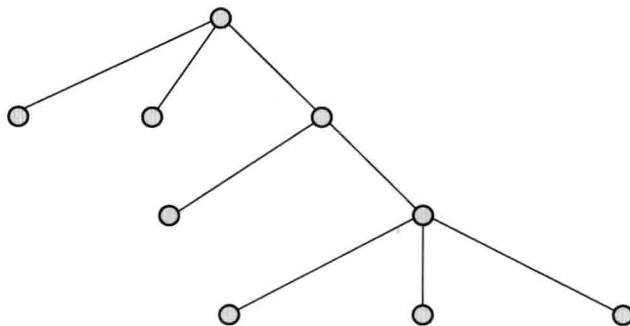


图 6-2 目录树结构

在 LDAP 目录中任何一个节点都可以包含信息, 同时也可以是一个容器, 也就是说任何一个 LDAP 条目都可以有子节点。图 6-4 表示了一个典型的目录结构, 在目录树中, 条目 ou=People, dc=example, dc=com 和条目 ou=Devices, dc=example, dc=com 都既包含属性又包含子节点。

通过命名模型, 可以给出目录中任何条目的唯一名称, 从而可以毫无歧义地引用任何一个条目。在 LDAP 中使用 DN 来引用条目。

在目录中, 按目录树从下到上对条目进行命名, 如图 6-4 中的灰色条目的 DN 名称为 uid=bjensen, ou=people, dc=example, dc=com。因为这种树形结构决定了从根节点到其他

任何一个节点的路径是唯一的，所以说每一个条目的 DN 是唯一的。

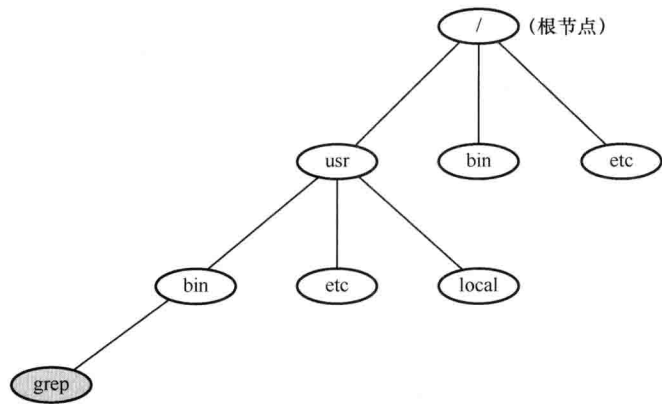


图 6-3 UNIX 文件系统

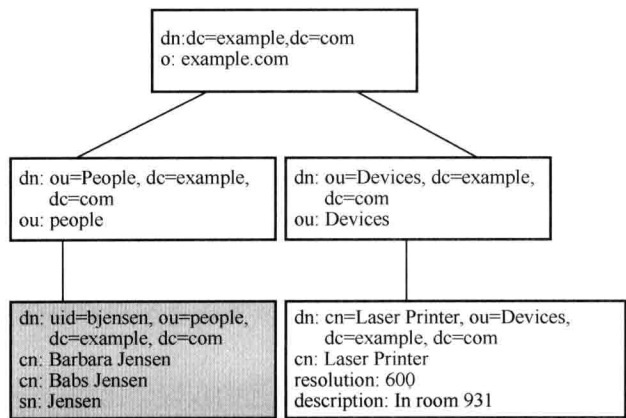


图 6-4 典型目录结构

在 DN 中最左边的内容称为相对辨别名 (RDN, Relative Distinguished Name)。如 ou=People, dc=example, dc=com 的 RDN 为 ou=People。对于共享同一个父节点的所有节点的 RDN 必须是唯一的。如果不属于同一个父节点，则节点的 RDN 可以相同。

当特殊字符出现在 DN 中时，必须进行转义，如表 6-2 所示。

表 6-2 特殊字符转义表

字符	数值	转义序列	字符	数值	转义序列
在 DN 或 RDN 开始或结尾的空格	32	\空格	反斜线 (\)	92	\\
在 DN 或 RDN 开始的#	35	\#	小于符 (<)	60	\<
逗号 (,)	44	\,	大于符 (>)	62	\>
加号 (+)	43	\+	分号 (;)	59	\;
双引号 (")	34	\"			

3. 功能模型

LDAP 功能模型描述了 LDAP 协议可以采用的相关操作，以访问存储在目录树中的数

据。在 LDAP 中共有 4 类 10 种操作。

- ① 查询类操作，如搜索、比较。
- ② 更新类操作，如添加条目、删除条目、修改条目、修改条目名。
- ③ 认证类操作，如绑定、解绑定。
- ④ 其他操作，如放弃和扩展操作。

除了扩展操作，另外 9 种是 LDAP 的标准操作。扩展操作是 LDAP 中为了增加新的功能提供的一种标准的扩展框架，当前已经成为 LDAP 标准的扩展操作，有修改密码和 StartTLS 扩展，在新的 RFC 标准和草案中正在增加一些新的扩展操作，不同的 LDAP 厂商也均定义了自己的扩展操作。

4. 安全模型

安全模型提供一个安全框架，保护目录中的信息不被非法访问。LDAP 中的安全模型主要通过身份认证、安全通道和访问控制（ACL）实现。

LDAP 是一个面向连接的协议，在能够对 LDAP 目录进行任何操作之前，LDAP 客户端必须获得一个到 LDAP 服务端的连接，在这个过程中需要对 LDAP 客户端的身份进行验证，这一过程可以理解为用户绑定。

LDAP v2 只支持简单的密码验证。LDAP v3 实现了 SASL 安全框架，SASL 为多种验证协议提供了一种标准的验证方法，对于不同的验证系统，可以实现特定的 SASL 机制。

在 LDAP 中提供了基于 SSL/TLS 的通信安全保障。SSL/TLS 基于 PKI 信息安全技术，LDAP 通过 StartTLS 方式启动 TLS 服务，可以提供通信中的数据保密性、完整性保护；通过强制客户端证书认证的 TLS 服务，同时可以实现对客户端身份和服务器端身份的双向验证。

在用户通过验证之后，可以为该用户分配附加的权限。比如，一些用户只能查看特定的条目，而不能修改；一些用户可以查看并且修改所有的条目等。这一过程可以理解为访问控制。

6.1.5 LDAP Schema

目录 Schema 是一个规则集，定义了 LDAP 目录所应遵循的结构和规则，它决定哪些信息可以存放在目录服务中，以及在客户端和目录服务器查询交互中如何处理信息。目录服务器在存储新数据或修改现有数据时，会检查数据是否满足 schema 规则，当客户端或服务器比较两个属性值时，它们会使用 schema 规定的比较算法。

如果使用过关系型数据库，那么对模式应该不会陌生。关系数据库系统都是通过表格的形式进行数据存储的。在这之前，要首先定义表结构，即模式。表结构由一些字段组成，每个字段都有一个类型，以及一些约束条件。这就规定了我们可以存储的信息。

不过与关系型数据库系统不同的是，作为 LDAP 目录服务器的用户而言，一般不需要自己定义模式，所有实现 LDAP 协议的目录服务器都已经定义好了许多模式，这些模式可以解决大部分的信息存储问题。很多常用 schema 都在 RFC2252 中进行了定义，主流 LDAP 服务器都会支持这些基本的 schema。

在 LDAP 的 schema 中，有 4 个重要元素：对象类、属性、语法、匹配规则。

1. 对象类 (objectClass)

objectClass 定义了一个对象类，它说明了该对象类有哪些属性，哪些属性是必需的，哪些又是可选的。一个对象类的定义包括名称、描述、类型（包括结构类型或辅助类型）、必需属性、可选属性等信息。

2. 属性 (attribute)

属性就是一个对象类中可能包含的属性，对其的定义包括名称、数据类型、单值还是多值以及匹配规则等。

3. 语法 (Syntax)

语法是 LDAP 中会用到的数据类型和数据约束，它遵从 X.500 中数据约束的定义。其定义需要有一个 ID（遵从 X.500）以及说明（DESP）。

4. 匹配规则 (Matching Rules)

下面列举出了一些 LDAP 规定好的或是现在比较通用的 schema，一般的 LDAP 服务器都应该可以识别这些定义。

下面是一个名为 subschema 的对象类的定义：

```
(2.5.20.1 NAME 'subschema' AUXILIARY
 MAY ( dITStructureRules $ nameForms $ ditContentRules $
 objectClasses $ attributeTypes $ matchingRules $ matchingRuleUse ))
```

首先是 OID，这里是 2.5.20.1，接着是 NAME 名为 subschema，AUXILIARY 说明是辅助型，之后是可选属性的定义，subschema 中没有定义必需属性，如果需要定义，应该和 MAY 一样，将属性放在 MUST () 中并用 \$ 隔开。

再来看一个属性 cn 的定义：

```
( 2.5.4.3 NAME 'cn' SUP name EQUALITY caseIgnoreMatch )
```

首先是 OID，这里是 2.5.4.3，接着是 NAME 名为 cn，再接着是父属性，名为 name，使用 caseIgnoreMatch 进行相等性 (EQUALITY) 匹配。

语法定义一般都比较简单，如：

```
( 1.3.6.1.4.1.1466.115.121.1.6 DESC 'String' )
```

这个定义说明，这一串数字 1.3.6.1.4.1.1466.115.121.1.5 代表了 LDAP 中的字符串，这个数字串的定义和 X.500 相关，包括了它的存储方式、所占空间大小等。

最后是匹配规则的例子，下面是 caseIgnoreMatch 的匹配规则：

```
( 2.5.13.2 NAME 'caseIgnoreMatch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)
```

首先是 OID，这里是 2.5.13.2，接着是 NAME 名 caseIgnoreMatch，再接着是语法，规定了数据类型，1.3.6.1.4.1.1466.115.121.1.15 就是 LDAP 数据类型 DirectoryString 的 OID，说明应用 'caseIgnoreMatch' 匹配规则的属性的类型必须是 DirectoryString 这个数据

类型才有效。

表 6-3 给出了 LDAP 协议中定义的一些常用属性及其含义(具体信息见 RFC 2252 文档)。

表 6-3 LDAP 协议定义的常用属性

属性	中文名称	描述
c	国家名称	值为两位国家代码, 例如, 中国: CN 美国: US
cn	通用名称	
dc	域名组件	如 sohu.com、dc=sohu, dc=com
co	国家名称	国家的全名
gn	givenName	
homephone	家庭电话号码	
mail	邮件地址	
mobile	移动电话号码	
o	组织名称	
ou	部门名称	通常为组织机构下的一个部门或者一个大型实体下的一个子实体
postalCode	邮政编码	
sn	姓, 别名	
st	州或者省的名称	
street	街道地址	
userPassword	用户密码	
uid	用户 ID	
departmentNumber	部门编号	
displayName	显示名称	
description	描述	
employeeNumber	员工编号	
manager	经理	

表 6-4 是一些 LDAP 协议中定义的对象类(具体信息查阅 RFC 2252 文档)。

表 6-4 LDAP 协议定义的常用对象类

对 象 类	必需属性	可 选 属 性
account	userid	description \$ seeAlso \$ localityName \$ organizationName \$ organizationalUnitName \$ host
country	c	searchGuide \$ description
dcObject	dc	
device	cn	serialNumber \$ seeAlso \$ owner \$ ou \$ o \$ l \$ description
inetOrgPerson->person 继承 person		audio \$ businessCategory \$ carLicense \$ departmentNumber \$ displayName \$ employeeNumber \$ employeeType \$ givenName \$ homePhone \$ homePostalAddress \$ initials \$ jpegPhoto \$ labeledURI \$ mail \$ manager \$ mobile \$ o \$ pager \$ photo \$ roomNumber \$ secretary \$ uid \$ userCertificate \$ x500uniqueIdentifier \$ preferredLanguage \$ userSMIMECertificate \$ userPKCS12

(续表)

对 象 类	必需属性	可 选 属 性
organizationalPerson 继承 Person		title \$ x121Address \$ registeredAddress \$ destinationIndicator \$ preferredDeliveryMethod \$ telexNumber \$ teletexTerminalIdentifier \$ telephoneNumber \$ internationaliSDNNumber \$ facsimileTelephoneNumber \$ street \$ postOfficeBox \$ postalCode \$ postalAddress \$ physicalDeliveryOfficeName \$ ou \$ st \$ l
organization	o	userPassword \$ searchGuide \$ seeAlso \$ businessCategory \$ x121Address \$ registeredAddress \$ destinationIndicator \$ preferredDeliveryMethod \$ telexNumber \$ teletexTerminalIdentifier \$ telephoneNumber \$ internationaliSDNNumber \$ facsimileTelephoneNumber \$ street \$ postOfficeBox \$ postalCode \$ postalAddress \$ physicalDeliveryOfficeName \$ st \$ l \$ description
organizationalRole	cn	x121Address \$ registeredAddress \$ destinationIndicator \$ preferredDeliveryMethod \$ telexNumber \$ teletexTerminalIdentifier \$ telephoneNumber \$ internationaliSDNNumber \$ facsimileTelephoneNumber \$ seeAlso \$ roleOccupant \$ preferredDeliveryMethod \$ street \$ postOfficeBox \$ postalCode \$ postalAddress \$ physicalDeliveryOfficeName \$ ou \$ st \$ l \$ description
organizationalUnit	ou	userPassword \$ searchGuide \$ seeAlso \$ businessCategory \$ x121Address \$ registeredAddress \$ destinationIndicator \$ preferredDeliveryMethod \$ telexNumber \$ teletexTerminalIdentifier \$ telephoneNumber \$ internationaliSDNNumber \$ facsimileTelephoneNumber \$ street \$ postOfficeBox \$ postalCode \$ postalAddress \$ physicalDeliveryOfficeName \$ st \$ l \$ description
Person	cn sn	userPassword \$ telephoneNumber \$ seeAlso \$ description
Top (所有类的基类)		

6.1.6 LDAP 认证方式

LDAP 提供多种认证策略, 包括匿名认证、简单口令、基于 SASL 的散列认证、基于 SSL 的简单口令、基于 SSL 的双向证书认证等。

① 匿名认证。在某些场景下, 需把数据共享给目录下的所有用户, 此时不需要对用户身份进行认证。

② 简单口令。由客户端向服务端发送身份和口令, 完成身份验证, 是最简单的认证方式, 但存在安全风险。

③ 基于 SASL 的散列认证。与简单口令相比, 客户端不向服务器发送口令, 而是服务器向客户端发送挑战码, 客户端以其知道的秘密信息对挑战码进行处理后返回给服务器端, 服务器据此判断用户身份。

④ 基于 SSL 的简单口令。与简单口令相比, 数据的传输在 SSL 通道上进行, 防止了口令的泄密。

⑤ 基于 SSL 的双向证书认证。基于 SSL 的数字证书认证具有最高的安全级别, 能够保证数据的保密性、数据的完整性, 并且利用数字签名技术对客户端身份进行验证。

在选择完认证策略后, 必须使用访问控制列表对用户可以访问的数据进行控制。每个访问控制列表指定三个方面的内容。

① 目录中的一个或多个资源: 资源又称为对象, 是访问控制的目标, 通常包括条目、

属性、属性值。

② 一个或多个主体：主体是拒绝或同意访问的条目，主体可以用目录条目的名称或描述性信息指定。

③ 一个或多个访问权限：访问权限决定主体能够操作的资源。例如，访问权限是“搜索和读取”，表明主体可查询并读取条目的属性。

由于不同目录服务的实现有差异，我们以 OpenLDAP 的访问控制为例。首先看一下 ACL 访问指令的格式：

```
<access directive> ::= access to <what> [by <who> [<access>] [<control>]]+
    <what> ::= * | [dn[.<basic-style>]=<regex> | dn.<scope-style>=<DN>]
        [filter=<ldapfilter>] [attrs=<attrlist>]
    <basic-style> ::= regex | exact
    <scope-style> ::= base | one | subtree | children
    <attrlist> ::= <attr> [val[.<basic-style>]=<regex>] | <attr> , <attrlist>
    <attr> ::= <attrname> | entry | children
    <who> ::= * | [anonymous | users | self
        | dn[.<basic-style>]=<regex> | dn.<scope-style>=<DN>]
        [dnattr=<attrname>]
        [group[/<objectclass>[/<attrname>][.<basic-style>]]=<regex>]
        [peername[.<basic-style>]=<regex>]
        [sockname[.<basic-style>]=<regex>]
        [domain[.<basic-style>]=<regex>]
        [sockurl[.<basic-style>]=<regex>]
        [set=<setspec>]
        [aci=<attrname>]
    <access> ::= [self]{<level>|<priv>}
    <level> ::= none | disclose | auth | compare | search | read | write | manage
    <priv> ::= {=|+|-} {m|w|r|s|c|x|d|0}+
    <control> ::= [stop | continue | break]
```

指令中包含一个 to 语句，多个 by 语句。这个指令的大体意思是，通过 access to 约束我们访问的资源，通过 by 设定哪个用户（who）有什么权限，并控制（control）这个 by 语句完成后是否继续执行下一个 by 语句或者下一个 ACL 指令。

资源有多种形式，如 DN、属性、过滤条件。

（1）通过约束 DN 访问

例如：access to dn="uid=matt, ou=Users, dc=example, dc=com" by * none

这个指令是指访问 uid=matt, ou=Users, dc=example, dc=com 这个 DN，即把访问的范围约束在这个 DN 中。by * none 是指对于任何人的访问都是拒绝的。

总体的意思就是，任何人都没有权限访问 uid=matt, ou=Users, dc=example, dc=com 这个 DN。当然，服务器管理员是可以访问的，不然就无法维护这个 OpenLDAP 中的用户信息。

再例如：access to dn.subtree="ou=Users, dc=example, dc=com" by * none

在这个例子中使用了 dn.subtree。在我们的目录信息树中，在 ou=Users 子树下可能有

多个用户。举例来说, DN 为 uid=matt, ou=Users, dc=example, dc=com 就是 ou=Users, dc=example, dc=com 的子树, 当试图要访问它时, 这个 ACL 指令就起了作用。

总体的意思是, 任何人没有权限访问 ou=Users, dc=example, dc=com 及其子树的信息。

DN 表示方式分别是:

① dn.base: 约束这个特定 DN 的访问。它和 dn.exact、dn.baselevel 是相同的意思。

② dn.one: 约束这个特定的 DN 第一级子树的访问。它和 dn.onelevel 是同义词。

③ dn.children: 和 dn.subtree 类似, 都是对其以下的子树访问权的约束。不同点在于, 这个约束是不包含自己本身的 DN, 而 subtree 包含了本身的 DN。

对于 dn 的约束条件还可以利用正则表达式, 如下:

```
access to dn.regex="uid=[^, ]+, ou=Users, dc=example, dc=com" by * none
```

这个指令将约束所有 uid=(任何值), ou=Users, dc=example, dc=com 的 DN, 其中的任何值是用[^,]+这个符号组合来表示的, 可以代表任何至少有 1 个字符且字符当中没有逗号(,)的值。更明确点说, 就是在 ou=Users, dc=example, dc=com 这个 DN 下所有以 uid 为属性的一级子树都属于这个约束的范围。

(2) 通过约束 attrs 访问

对于 DN 的约束大多用在对某个层级的约束, 而使用 attrs 就可以跨层级(或者跨越父类树), 通过属性来约束访问的范围。

```
access to attrs=homePhone by * none
```

这个例子的意思是: 任何人没有权限访问属性为 homePhone 的信息。

在 attrs 后面的值可以有多个, 例如:

```
access to attrs=homePhone, homePostalAddress
```

如果要约束某个对象类的所有属性, 可以采用以下形式:

```
access to attrs = title, registeredAddress, destinationIndicator, ...
```

但这个方法太耗时, 也难以阅读, 显得笨重, 以下给出一个好的方法:

```
access to attrs=@organizationalPerson by * none
```

采用@的方法必须谨慎, 这段指令不仅仅约束了 organizationalPerson 里的属性, 也约束了 person 对象类的属性。为什么? 因为 organizationalPerson 对象类是 person 的子类, 因此, 所有 person 中的属性当然也是 organizationalPerson 的属性。

如果想做除了 organizationalPerson 的其他对象类的约束, 可以用“!”来表示:

```
access to attrs=!organizationalPerson
```

也可以加入属性的值, 具体约束某个值:

```
access to attrs=givenName val="Matt"
```

这个指令也可以采用正则表达式约束的方法, 如下:

```
access to attrs=givenName val.regex="M.*"
```

最后给出一个一般情况下用到的利用属性约束的例子：

```
access to attrs=member val.children="ou=Users, dc=example, dc=com" by * none
```

(3) 通过过滤 (filter) 访问

过滤提供一种支持条目记录匹配的方法，如下：

```
access to filter="(objectClass=simpleSecurityObject)" by * none
```

这表示可以约束所有记录中包含对象类为 simpleSecurityObject 的信息。

与编程语言类似，ACL 指令也有类似与或的条件判断，如下：

```
access to filter="(!((givenName=Matt)(givenName=Barbara))(sn=Kant))" by * none
```

这段代码过滤出 givenName 为 Matt 或 Barbara 或者 surname 为 Kant 的信息。

6.1.7 LDIF 数据交换文件

LDAP 数据交换格式 (LDAP Data Interchange Format) 是在 RFC2849 中定义的基于文本描述目录条目的一种标准格式，即使两个服务器使用不同的内部数据存储格式，LDIF 也可以导出目录数据并将其导入到另一个目录服务器。

LDIF 文件是文本文件，可以由 5 种类型的行构成：指令行、续行、空行、注释行、分隔行。空行就是不包含任何字符的行；注释行是以井号 (#) 开头的行；分隔行是只有减号 (-) 的行，它用来分隔对一个目录条目的多个操作；续行是以一个空格开头的行；指令行是除了以 # 号和空格开头的行。

有两种不同类型的 LDIF 文件。第一类描述了一组目录条目，如整个企业目录，或者是企业目录的一个子集；另一种类型的 LDIF 文件是一系列的目录条目更新语句，用于更新现有的目录条目数据，在 RFC 2849 中有完整格式的定义。下面分别说明。

6.1.7.1 第一种类型文件

第一种类型文件的内容包含两部分：第一部分是 DN，第二部分是系列的属性值对。如下所示：

```
dn: uid=bjensen, ou=people, dc=example, dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: Barbara Jensen
cn: Babs Jensen
givenName: Barbara
sn: Jensen
uid: bjensen
mail: bjensen@example.com
telephoneNumber: +1 408 555 1212
description: Manager, Switching Products Division
```

DN 必须是条目的第一行，由字符串“DN”后跟一个冒号(:)和条目的区别名组成。DN 后面是条目的属性，每个属性由一个属性类型、一个冒号(:)和属性值组成。属性可以以任何顺序出现，以增强可读性。但是一般先列出条目的对象类，并把相同属性类型的值放在一起。

当一行数据很长时，通常的做法是对数据进行换行。LDIF 文件支持换行，具体做法是：在折行的地方插入一个换行符和空格符。如下所示：

```
description: I will be out of the
            office from August 12, 2001, to September 10, 2001.
```

如果 LDIF 中的属性值不为 ASCII，则必须用 Base64（参考）进行编码，采用 Base64 编码的值用两个冒号(::)分隔属性和值，如：

```
jpegPhoto:: /9j/4AAQSkZJRgABAAAAQABAAD/2wBDABALDA4MChAODQ4
SERATGCgaGBYWGDEjJR0oOjM9PDkzODdASFxOQERXRTc4UG1RV19iZ2hnP
```

6.1.7.2 第二种类型文件

第二种类型文件包含更新语句。第一行同样是 DN，第二行是更新类型，后面是要更新的属性及值，也可以用来添加新的条目。

1. 增加条目

changetype 类型 add 表示增加一个条目到目录中，格式为：

```
dn: 要增加条目的 DN 值
changetype: add
attribute type: value
```

2. 删除条目

changetype 类型 delete 表示从目录中删除一个条目，格式为：

```
dn: 要删除条目的 DN 值
changetype: delete
```

3. 修改条目

changetype 类型 modify 表示修改目录中的一个条目，可以增加新的属性值，删除指定属性值，删除全部属性，替换属性值为新值，格式为：

```
dn: 要修改条目的 DN 值
changetype: modify
modifytype: attribute type
[attribute type: attribute value]
```

可以看到上面增加了新的操作符 modifytype，它的值可为 add、delete 或 replace，示例如下：

① 增加 telephoneNumber 属性：

```
dn: uid=bjensen, ou=people, dc=example, dc=com
```

```
changetype: modify
add: telephoneNumber
telephoneNumber: +1 216 555 1212
telephoneNumber: +1 408 555 1212
```

② 删除 telephoneNumber 属性的+1 216 555 1212 值:

```
dn: uid=bjensen, ou=people, dc=example, dc=com
changetype: modify
delete: telephoneNumber
telephoneNumber: +1 216 555 1212
```

③ 完全删除 telephoneNumber 属性值:

```
dn: uid=bjensen, ou=people, dc=example, dc=com
changetype: modify
delete: telephoneNumber
```

④ 替换 telephoneNumber 属性值为 2 项新值:

```
dn: uid=bjensen, ou=people, dc=example, dc=com
changetype: modify
replace: telephoneNumber
telephoneNumber: +1 216 555 1212
telephoneNumber: +1 405 555 1212
```

⑤ 多个操作可以组合起来使用，中间用单独一行“-”进行分隔，如:

```
dn: uid=bjensen, ou=people, dc=example, dc=com
changetype: modify
add: mail
mail: bjensen@example.com
-
delete: telephoneNumber
telephoneNumber: +1 216 555 1212
-
delete: description
-
replace: givenName
givenName: Barbara
givenName: Babs
-
```

4. 重命名或移动条目

重命名或移动条目可使用值为 moddn 的 changetype 操作，修改了 DN，可把条目移动到目录树的新位置。

```
dn: 要修改条目的 DN 值
changetype: moddn
[newsuperior: 新的父条目 DN]
[deleteoldrdn: ( 0 | 1 )]
[newrdn: 条目新 RDN]
```

如果修改 RDN，则必须提供 newrdn 和 deleteoldrdn 参数；如果把条目移动到新位置，则必须提供 newsuperior 参数。修改 RDN 示例如下：

```
dn: uid=bjensen, ou=People, dc=example, dc=com
changetype: moddn
newrdn: uid=babsj
deleteoldrdn: 0
```

移动条目到新位置示例：

```
dn: uid=bjensen, ou=People, dc=example, dc=com
changetype: moddn
newsuperior: ou=Terminated Employees, dc=example, dc=com
```

6.2 常见 LDAP 产品介绍

6.2.1 IBM TDS

IBM Tivoli 目录服务器 (ITDS) 的前身为 IBM 目录服务器，实现了轻量级目录访问协议 LDAP，是 IBM 的 Tivoli 身份与访问管理产品的一部分。ITDS 可以跨平台进行安装配置。

ITDS 提供了以下组件：一个使用 DB2 数据库对目录信息进行存储的服务器，一个将 LDAP 操作路由到其他服务器上的代理服务器，一个客户端，一个管理服务器的图形界面，一个管理用户的图形界面。

ITDS 支持多种身份验证方式，包括：用户名和密码验证、数字证书身份验证、简单验证和安全层 (SASL)、挑战-响应身份验证机制 (CRAM-MD5)、Kerberos 身份验证。

ITDS 具有如下特点：

- ① 支持数百万目录条目。使用 IBM DB2 技术存储目录数据。
- ② 实现 LDAP 相关规范。提供符合 LDAP 规范的按需应变的身份基础设施。
- ③ 利用强大的主/从和对等复制提供高可用性，支持多达数十个主服务器。
- ④ 集成了 IBM 中间件、身份管理和安全产品，支持与非 IBM 产品集成。
- ⑤ 支持基于 Web 进行系统管理。
- ⑥ 支持主流平台，包括 AIX、Solaris、Windows Server、HP-UX、SUSE、Red Hat。

6.2.2 Sun Java 系统目录服务器

Sun Java 系统目录服务器 (Sun Java System Directory Server) 早期称为 Sun ONE 目录服务器、iPlanet 目录服务器，更早之前称为 Netscape 目录服务器。它包括 Sun LDAP 目录服务器和 DSML 服务器，是 Java 企业系统的一个组件。

在 Sun 被 Oracle 收购后, Sun Java 系统目录服务器成为 Oracle 目录服务器企业版 (ODSEE) 的一部分。

Sun 目录服务支持 LDAP v2 和 LDAP v3 相关 RFC 标准, 包括: RFC2079, RFC2246, RFC2247, RFC2307, RFC2713, RFC2788, RFC2798, RFC2831, RFC2849, RFC2891, RFC3045, RFC3062, RFC3296, RFC3829, RFC3866, RFC4370, RFC4422, RFC4505, RFC4511, RFC4512, RFC4513, RFC4514, RFC4515, RFC4516, RFC4517, RFC4519, RFC4522, RFC4524, RFC4532。

Sun 目录服务器支持在以下平台上安装运行: Solaris 9 和 10 操作系统、OpenSolaris、Red Hat Enterprise Linux、SuSE Linux、HP-UX、Windows Server。

6.2.3 Novell eDirectory

Novell 公司的 eDirectory (前身为 Novell 目录服务, 有时也称为 NetWare 目录服务) 是一个兼容 X.500 的目录服务软件产品, 最初发布于 1993 年。eDirectory 是层次化的面向对象的数据库, 它用逻辑树的方式表示组织资产, 这些资产包括: 组织机构、组织单位、人、位置、服务器、卷、工作站、应用程序、打印机、服务和团体。eDirectory 是高度可伸缩的高性能安全目录服务, 支持安全套接层 (Secure Socket Layer, SSL) 上的 LDAP v3 协议, 提供复制和分区功能。

eDirectory 中使用权限动态继承机制, 支持全局和局部访问控制。目录树中对象访问权限在被访问时确定, 每个对象可以依据在目录树中的位置、安全等级、个体赋值等确定访问权限。该软件支持在树中的任何一点上进行分区, 以及复制任何分区到任意数量的服务器上。服务器之间周期性地增量复制。每个服务器都可以充当主服务, 向其他服务器赋值它拥有的信息。此外, 可以设置从服务器只接收定义的属性以提高复制速度 (例如, 可以配置一个从服务器, 其只包括公司地址簿上的姓名和电话号码, 而不是使用整个目录的用户配置文件)。

eDirectory 支持引用完整性、多主复制、并具有模块化的认证架构。它支持通过 LDAP、DSML、SOAP、ODBC、JDBC、JNDI 和 ADSI 访问。

eDirectory 服务器支持在以下平台上安装运行: Novell NetWare、Solaris、Red Hat Enterprise Linux、SuSE Linux、HP-UX、Windows Server、IBM AIX。

6.2.4 GBase 8d

南大通用目录服务系统是原南开创元目录服务 (ITEC-iDS) 系统的升级版, 其核心产品 GBase 8d 已经广泛应用于我国省级 CA 和部委级 CA 等 PKI/PMI 系统中, 以及大型企事业单位的身份标识管理系统中, 并在省市级的电子政务建设中得到应用。

GBase 8d 是南大通用目录服务产品线的核心和基础, 其主要由 LDAP Server 和 Slurpd (镜像复制) Server 以及相关的管理软件和应用开发套件构成。

GBase 8d 具有以下特性:

- ① 遵循轻型目录访问协议 LDAP v2 和 v3。
- ② 支持 Windows 2000/2003、Linux、Solaris、AIX、HP-UX 操作系统。
- ③ 单服务器管理容量可达千万级条目。

- ④ 支持 LDIF 格式的导入/导出。
- ⑤ 支持 RFC 规范定义的分页标准（规范定义的分页标准 RFC2696）。
- ⑥ 支持匿名、用户名/简单密码、用户名/摘要密码等多种身份认证方式。
- ⑦ 对于主体的授权，可基于主体的位置特征、主体的属性特征进行策略授权。
- ⑧ 支持 SSL 和 TLS 实现的安全通道，实现信息传递的私密性保护。
- ⑨ 支持全库统计条目及子树统计条目。
- ⑩ 支持 Schema 的扩展，可实现用户自定义 Schema 文件的载入，用户自定义对象类和属性的加入。
- ⑪ 支持 LDAP 的引用（referral）功能。
- ⑫ 提供图形化的管理界面，管理员可同时管理多个目录服务器。
- ⑬ 使用中文作为 DN，支持 UTF-8 和 GB 两套汉字编码体系。

6.2.5 OpenLDAP

OpenLDAP 项目开始于 1998 年，是轻型目录访问协议的开源实现，在 OpenLDAP 许可证下发行，并已经包含在众多流行的 Linux 发行版中。OpenLDAP 可以运行于 BSD 变种操作系统、AIX、Android、HP-UX、Mac OS X、Solaris、Windows 等操作系统。

OpenLDAP 项目在 2007 年 10 月发布了 OpenLDAP 2.4 版本，引入了多主复制、热备主服务、动态删除和修改 Schema 元素等特性。

OpenLDAP 软件包含 3 个主要组件：

- ① slapd：LDAP 守护进程及相关模块和工具。
- ② 库：实现 LDAP 协议和 ASN.1 基本编码规则（BER）。
- ③ 客户端软件：ldapsearch、ldapmodify、ldapmodrdn 及其他。

OpenLDAP 服务器在系统架构上分为前端和后端两个部分，前端负责网络访问和协议处理，后端负责数据存储处理。由于采用模块化技术，OpenLDAP 支持多种后端存储模式，包括数据存储后端、代理后端、动态后端。

- ① 数据存储后端：进行实际数据存储，如 back-bdb、back-ldif 等。
- ② 代理后端：充当其他存储系统的网关，如 back-ldap、back-sql 等。
- ③ 动态后端：动态产生数据如 back-config、back-sock 等。

OpenLDAP 支持使用在 RFC4533 中指定的内容同步复制机制，除了基本规范，也支持称为 delta-syncrpl 的增强功能。其他增强功能已经实现，支持多主复制。欲了解复制的更多信息，请参阅 RFC4533。

6.2.6 Microsoft Active Directory

活动目录（Active Directory）是微软 Windows Server 中负责构建中大型网络环境的集中式目录服务，从 Windows 2000 Server 开始内置于 Windows Server 产品中，它处理了组织中的网络对象（对象可以是用户、组群、计算机、域名控制站、邮件、设置文件、组织单元、树系等），只要是在活动目录 Schema 中定义的对象，就可以存储在活动目录数据文件中，并利用活动目录服务接口（Service Interface）来访问。实际上，许多活动目录的管

理工具都利用这个接口来调用并使用活动目录的数据。

活动目录最早在 1996 年出现，并在 Windows 2000 中首次问世，研发代号为 Cascade，并历经 Windows 2000、Windows Server 2003 的演化，目前活动目录已成为成熟的目录服务组件，在 Windows Server 2008 中，活动目录服务更将其角色扩充至 5 种服务（包含网络目录服务、凭证服务、联邦服务、轻量级服务、权限管理服务 etc.）。

6.3 LDAP 部署与优化

6.3.1 复制介绍

通过把目录中的数据放在多个位置，可以提高目录服务的可靠性。如果一台服务器出现故障，目录客户端和应用程序可以与不同目录服务连接。通过复制机制，提高了应用系统的可用性。使用目录复制的内容，多个客户端目录数据请求可以负载到多个服务器上，从而改善目录服务的性能。

在复制系统中，我们使用提供者和服务消费者分别代表数据源和目的地，提供者服务器发送更新到另一台服务器，消费者服务器接收这些变化。提供者和消费者角色不是相互排斥的，一个服务器可能既是提供者又是消费者。

提供者和消费者的服务器的配置信息称为复制协议。此配置信息通常包括复制的单元，主机名和远程服务器的端口，以及复制间隔等其他信息。复制协议描述了哪个消费者应该接收更新、目录哪一部分被复制、该如何连接提供者和服务消费者，以及如何验证消费者。

在大多数目录服务器软件中，目录分区的单元也是复制的单元，所以分区方案决定了复制单元。一些目录服务器软件允许创建目录子集的副本，只有目录的一部分复制到消费者服务器中。

一致性程度说明在任何时刻，提供者和服务消费者之间数据的一致程度。强一致性复制指在任何时刻，提供者和服务消费者的数据是一致的。弱一致性复制允许消费者服务器数据和服务提供者服务器数据在一段时间内有偏离，但数据最终会趋向一致。

目录复制支持增量更新和全部更新，增量更新指消费者服务器只更新变化部分，全部更新指消费者服务器同步所有数据。在初始化复制时，必须进行全复制，保证消费者与服务提供者数据的一致性。

一般情况下，目录的整个分区都复制到消费者，如图 6-5 所示，子树 `ou=Accounting, dc=example, dc=com` 被完整复制到消费者服务器上。

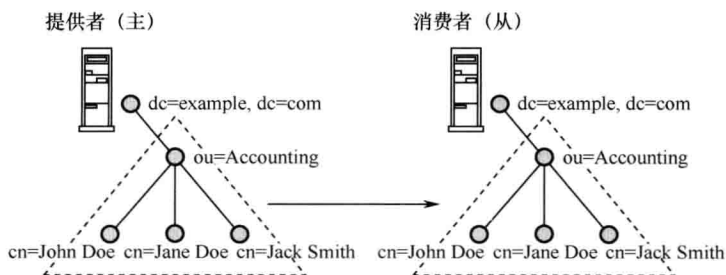


图 6-5 复制条目子树

当然，在某些情况下，只需要复制条目的部分属性，如图 6-6 所示，防火墙之外的副本中 John Doe 的条目包含了比防火墙内的主服务更少的属性。

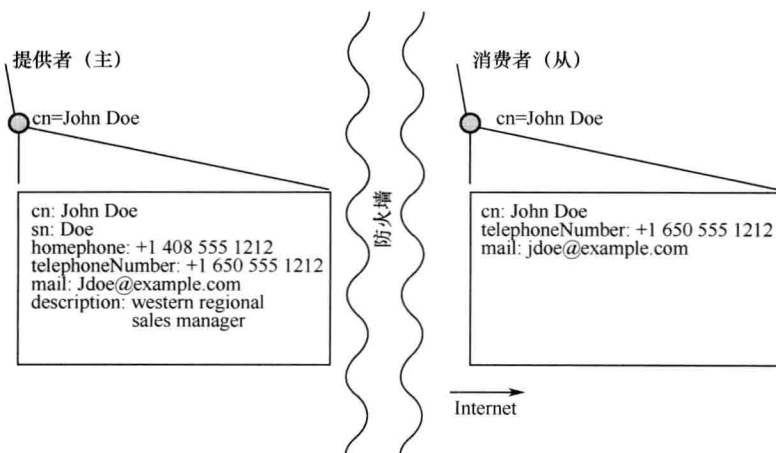


图 6-6 部分属性复制

在某些情况下，我们可能只需要复制子树下的部分条目，如图 6-7 所示，使用对象类选择复制条目，只有子树 `ou=Accounting, dc=example, dc=com` 下的组织单元(organizational Unit)和个人(person)条目被复制。

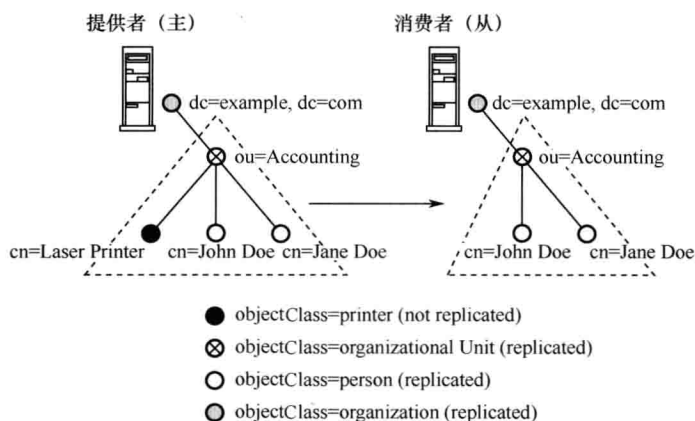


图 6-7 部分子树复制

目录复制方式主要包括一主多从或多主复制，一主多从表示只有一个主目录(提供者)，只有主目录可写，从目录只读，所有对从目录的写都会重定向到主目录。多主复制是指配置多个目录服务充当主目录的角色，每个主目录都可写入，写入的数据会同步到其他主目录。由于多主复制容易造成数据的不一致，所以在实际部署中多采用一主多从部署模式。

6.3.2 引用机制介绍

引用可以看作是一个别名，别名包含另一个对象的 DN，而引用包含一个或多个对象的 URL，通常情况此 URL 是 LDAP 的 URL。LDAP URL 中包含服务器的主机/端口和一个

对象的 DN，主机/端口的信息可以指向不同的目录服务器。别名解析由服务器处理，引用返回给客户端，由客户端负责处理它。

引用是用不同的名称来标识一个对象，它们允许目录管理员设置“搜索路径”，用于收集来自多个服务器的结果。它们还可以用于部署高速缓存或只读服务器副本返回所有更新请求的引用。可以使用引用实现只读副本的负载均衡策略。

例如，服务器 A 包含"DC=example, DC=net"，服务器 B 包含"DC=sub, DC=example, DC=net"，服务器 A 含有引用对象"DC=sub, DC=example, DC=net"，它的 ref 属性值为"ldap://B/DC=sub, DC=example, DC=net"。

```
dn: DC=sub, DC=example, DC=net
dc: sub
ref: ldap://B/DC=sub, DC=example, DC=net
objectClass: referral
objectClass: extensibleObject
```

一般情况下，引用对象和被引用对象的 DN 相同，如果 ref 有多个值，在 LDAP URL 中的 DN 值应该一致。管理员在配置目录服务时需要避免引用循环。

6.3.3 复制机制的部署

不同厂家的目录服务器在复制机制实现上会有差别。我们以 OpenLDAP 2.4 版本为例，说明复制机制的部署。

OpenLDAP 2.4 版本实现了 RFC4533 (LDAP Content Synchronization Operation) 复制技术，称为 syncrepl，对目录树的所有写入进行跟踪，对副本的写操作被拒绝，并返回主服务引用。

syncrepl 从服务器启动，现在将其命名为消费者，主服务器角色称为提供者。在 syncrepl 中，消费者连接到提供者以更新目录树。在最基本的仅刷新 (refreshOnly) 模式中，消费者接收自上一次更新以来的所有更改条目，请求状态标记 (cookie) 跟踪上一次同步的更改，然后断开连接。在下次连接时，将状态标记呈现给提供者，它仅发送自上一次同步之后更改的条目。

另一个 syncrepl 模式称为更新并保持 (refreshAndPersist)，如 refreshOnly 操作一样启动；但是不会断开连接，消费者保持连接以接收任何更新。在最初更新后发生的任何更改都会立即通过连接由提供者发送到消费者。

提供者的配置如下：

```
overlay syncprov
syncprov-checkpoint 100 10
syncprov-sessionlog 100
```

syncprov-checkpoint 100 10 告诉服务器每 100 次写操作或每隔 10 分钟将 contextCSN 的值存储到磁盘中。contextCSN 是 cookie 的一部分，它可以帮助消费者找到自上一个复制周期之后的某个位置。syncprov-sessionlog 100 的意思是将写操作存储到磁盘中，100 表示最大会话日志的数量。

基于 refreshOnly 模式的消费者配置如下：

```
updateref ldap://masterserver.ertw.com
syncrepl rid=1
provider=ldap://masterserver.ertw.com
type=refreshOnly
interval=00:01:00:00
searchbase="dc=ertw, dc=com"
bindmethod=simple
binddn="uid=replica1, dc=ertw, dc=com"
credentials=replica1
```

syncrepl 命令需要 updateref、尝试复制的目录树的信息，以及将要使用的验证凭证。凭证在消费者一方执行，并需要足够权限来读取正被复制的目录树部分，rid 将此消费者标识给主服务器。消费者必须是具有介于 1 到 999 之间的唯一 ID。provider 是指向提供者的 LDAP URI。type 指定只想通过 refreshOnly 进行定期同步，且 interval 是每小时。interval 以 DD:hh:mm:ss 格式指定。

转换到 refreshAndPersist 模式十分简单，只是移除 interval，并将 type 更改为 refreshAndPersist，如下：

```
updateref ldap://masterserver.ertw.com
syncrepl rid=1
provider=ldap://masterserver.ertw.com
type=refreshAndPersist
searchbase="dc=ertw, dc=com"
bindmethod=simple
binddn="uid=replica1, dc=ertw, dc=com"
credentials=replica1
```

当然，不必复制整个 LDAP 目录树，可以通过命令筛选只需复制的数据，筛选条件说明如表 6-5 所示。与 syncrepl 的其他选项一样，这些选项以 key=value 的形式输入。

表 6-5 复制筛选条件项

条 件	说 明
searchbase	指向复制将开始的树节点的 DN
scope	sub、one 或 base 之一。它确定从 searchbase 开始，到树下多深的数据将被复制，默认值为 sub，它涵盖 searchbase 和所有子 searchbase
filter	LDAP 搜索过滤器，例如 (objectClass=inetOrgPerson)，用于控制复制哪些记录
attrs	将从所选条目中复制的属性列表

一个部分复制的例子如下：

```
syncrepl rid=123
provider=ldap://provider.example.com:389
type=refreshOnly
```

```

interval=01:00:00:00
searchbase="dc=example, dc=com"
filter="(objectClass=organizationalPerson)"
scope=sub
attrs="cn, sn, ou, telephoneNumber, title, l"
schemachecking=off
bindmethod=simple
binddn="cn=syncuser, dc=example, dc=com"
credentials=secret

```

在这个例子中，消费者将从 `ldap://provider.example.com` 的 389 端口连接到提供者来执行每天一次 `refreshOnly` 模式的同步。它将以 `cn=syncuser, dc=example, dc=com` 绑定用户名，以密码“secret”进行简单验证。注意要在提供者服务器为 `cn=syncuser, dc=example, dc=com` 设置适当的访问控制权限以接收想要的复制内容，同步在 `dc=example, dc=com` 的整个子树搜索 `objectClass` 是 `organizationalPerson` 的条目，请求的属性是 `cn、sn、ou、telephoneNumber、title、l`。`schema` 检查被关闭，这样当处理来自提供者的更新时，消费者将不会强制对条目进行 `schema` 检查。

6.3.4 引用机制的部署

如前所述，引用的主要用途是建立分布式目录系统、对目录进行分区，或把多个小的目录系统组合成一个大的虚拟目录。

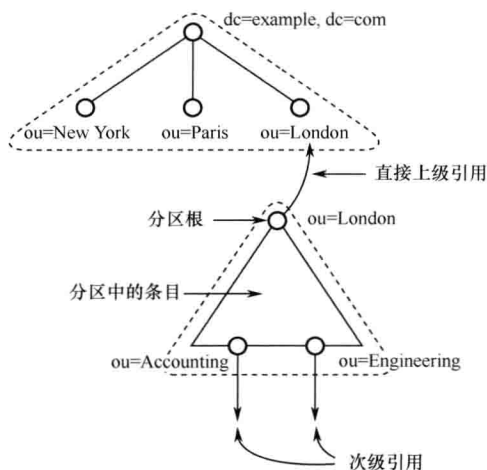


图 6-8 目录数据分区

如图 6-8 所示，该分区根 `ou=London, dc=example, dc=com` 代表分区的顶部。分区包含分区根目录和它下面的所有条目，除了在 `ou=Accounting, ou=London, dc=example, dc=com` 和 `ou=Engineering, ou=London, dc=example, dc=com` 分区包含的条目，也包含了指向总目录树的引用。当在一个分区搜索非本分区的条目时，会返回总体目录的引用。

在目录服务器中，向上引用一般配置在服务器的配置文件中，不出现在目录条目中。

例如，假设服务器 `hosta` 包含“`O=MNN, C=WW`”和“`CN=Manager, O=MNN, C=WW`”条目及以下引用对象：

```

dn: OU=People, O=MNN, C=WW
ou: People
ref: ldap://hostb/OU=People, O=MNN, C=US
ref: ldap://hostc/OU=People, O=MNN, C=US
objectClass: referral
objectClass: extensibleObject

```

```
dn: OU=Roles, O=MNN, C=WW
ou: Roles
ref: ldap://hostd/OU=Roles, O=MNN, C=WW
objectClass: referral
objectClass: extensibleObject
```

第一个引用对象告诉服务器 hosta: 服务器 hostb 和 hostc 拥有子树“OU=People, O=MNN, C=WW”，第二个引用对象表明服务器 hostd 拥有子树 “OU=Roles, O=MNN, C=WW”。

6.3.5 LDAP 优化

对 LDAP 性能的优化主要包括两个方面，采用索引、缓存技术提高单台服务器的查询性能；采用复制技术使多台服务器同时提供查询服务，以提高系统整体响应能力。

目录服务器采用的缓存技术随产品的不同而不同，如 OpenLDAP 使用配置文件设置服务器的缓存参数，在 slapd.conf 中使用“`cacheSize` 条目数”设置内存中缓存的条目数目，为了提高性能，可以设置缓存尽可能多的条目，或根据实际情况，设置合适的缓存数目，一种极端情况是把所有条目都缓存起来（只有在条目数目较少时可行）。OpenLDAP 使用 DBD 保存数据，通过设置 DBD 的缓存参数，可以优化读写性能，通过“`set_cachesize <gbytes> <bytes> <ncache>`”参数，设置缓存大小，`<gbytes>`加`<bytes>`的值越大，越能减少读写操作访问硬盘的次数。

通过对条目的属性建立索引，能够极大提高查询性能，索引方式随 LDAP 产品的不同而有区别。在 OpenLDAP 中，通过 `index` 语句在配置文件中建立属性索引。`index` 的格式为：

```
index attrlist | default indices
```

`indices` 的取值可为 `pres`、`approx`、`eq`、`sub`、`special` 中的一项或多项。`pres` 在使用“`objectclass=person`”或“`attribute=mail`”格式查询时需要；`approx` 在使用“`sn~=perso`”格式查询时必须指定。`eq` 在使用“`sn=smith`”格式时需要，特别是在使用 EQUALITY 规则而查询条件中没有通配符查询时；`sub` 在使用“`sn=sm*`”格式时需要，特别是在查询条件中有通配符时；`special` 值为 `nolang` 或 `nosubtypes`，与 `subtypes` 有关。

使用 `default` 指定缺省匹配规则，用在 `index` 语句中没有指定规则时，例如：

```
index default pres, eq
index cn, sn, uid
```

例如，对 `cn`、`sn`、`uid` 建立索引：

```
index cn pres, eq
index sn pres, eq
index uid pres, eq
```

当单台服务器性能不能满足查询需求时，就需要使用多台服务器对查询进行负载均衡了。使用负载均衡时，一般采用复制方式，所有的写操作都重定向到主服务器，所有的读操作优先使用从服务器。采用的原则是就近提供服务，即在一个分布式服务环境中，所有操作优先访问本地服务。如果本地服务不能提供，则访问中心服务。

6.4 面向 LDAP 的系统设计与开发

6.4.1 LDAP 管理工具

大多数商业和开源的 LDAP 实现会随软件带有目录管理工具，以对目录进行查询和操作，这些工具能够满足大部分目录操作的需求。为了方便用户操作，我们以 OpenLDAP 2.4 版本为例展示这些工具。

从 <http://www.userbooster.de/en/download/openldap-for-windows.aspx> 下载最新 Windows 版本的 OpenLDAP，根据提示进行安装。

从应用角度来说，对目录的操作无非是增加（ldapadd）、删除（ldapdelete）、修改（ldapmodify）、查询（ldapsearch）条目。ldapadd、ldapmodify、ldapdelete、ldapsearch 是各个 LDAP 产品都支持的命令行工具，它们有比较一致的命令行参数和使用约定。

虽然 ldapadd 和 ldapdelete 实现了条目的增加和删除，但 ldapmodify 实现的功能更多，用 ldapmodify 完全可以替代 ldapadd 和 ldapdelete 的功能，可以说 ldapadd 和 ldapdelete 是 ldapmodify 的简化定制版本，所以我们主要介绍 ldapmodify 和 ldapsearch 两个命令工具。

1. ldapmodify 命令

ldapmodify 的用法：ldapmodify [选项]

操作列表读取自 stdin 或通过 -f file 选项读取自文件。具体选项请参考表 6-6 和表 6-7。

表 6-6 ldapmodify 增加或修改选项

选项名称	含 义 说 明	选项名称	含 义 说 明
-a	向目录中增加属性值，缺省情况是替换属性值	-P	使用的协议版本，缺省为 v3
-c	遇到错误跳过，并继续执行	-S file	把跳过执行的操作写到文件 file 中
-f file	从文件 file 读取操作指令		

表 6-7 ldap 工具通用选项

选 项 名 称	含 义 说 明
-d level	将 LDAP 调试级别设置为“level”
-D binddn	绑定 DN
-h host	LDAP 服务器
-p port	LDAP 服务器上的端口
-H URI	指定要查询的服务器 URI（如：ldap://localhost:3389/），常见格式为 ldap(s)://hostname:port，如果使用了 -H，就不能使用 -h 和 -p 参数
-I	使用 SASL 交互模式
-n	显示该做而没有完成的操作
-O props	SASL 安全参数
-v	详细日志模式，诊断信息都输出到标准输出
-V	输出版本信息
-w passwd	简单认证模式下绑定口令
-W	提示输入绑定口令
-x	使用简单认证
-y file	从文件 file 读取口令
-Z	启用 TLS 请求（-ZZ 要求必须得到成功响应）

ldapmodify 可以一次执行目录服务器上的一个或多个更新, ldapmodify 依据 LDIF 文件依次执行更改操作, 该 LDIF 文件除了要遵循一般的 LDIF 文件格式外, 还要注意数据的次序, 比如要确保父节点在子节点之前, 用户密码只要写明文即可。虽然 ldapmodify 参数较多, 但常用的形式还是比较简单的, 如下:

```
ldapmodify -a -h host -p port -D <bind dn> -w <password> -f <ldif file>
```

假定要把 Jensen 的 e-mail 地址改为 abc@example.com, 可以采用命令 “ldapmodify -h localhost -D "cn=directory manager" -w secret -f updates.ldif”, updates.ldif 的内容为:

```
dn: uid=jensen, ou=people, dc=example, dc=com
changetype: modify
replace: mail
mail: abc@example.com
```

增加一个条目可以用命令 “ldapmodify -h localhost -D "cn=directory manager" -w secret -a -f updates.ldif”, updates.ldif 的内容为:

```
version: 1
dn: uid=bjensen, ou=people, dc=example, dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: Babs Jensen
givenName: Barbara
sn: Jensen
uid: bjensen
mail: bjensen@example.com
telephoneNumber: +1 408 555 1212
description: Manager, switching products division
```

从上面的例子看出, 修改条目属性操作和增加条目操作的命令行参数基本一致, 只是 LDIF 文件的内容不同, 对于其他 LDIF 操作示例, 参考 6.1.7 节 “LDIF 数据交换文件”。

2. ldapsearch 命令

与 ldapmodify 类似, ldapsearch 也是从命令行接收查询参数, 然后以 LDIF 格式显示搜索结果。ldapsearch 工具的用法如下:

```
ldapsearch [选项] [过滤条件 [属性列表]]
```

其中, “过滤条件” 符合 RFC4515 规定的查询过滤规则, “属性列表” 包含以空格分隔的属性说明, 其中 “*” 表示全部属性, “+” 表示操作属性, “1.1” 表示忽略属性。具体选项及过滤条件请参见表 6-8 和表 6-9。

与 ldapmodify 相同, ldapsearch 同样支持表 6-7 所列的选项。

当搜索条件所含表 6-10 中的字符时, 必须进行转义。

表 6-8 搜索选项列表

搜索选项	说 明
-a deref	指定别名反向引用。可选值为 never、always、search 或 find 如果不使用此参数，默认为 never 当服务端遇到别名时，它可选择反向引用或直接返回别名，“find”在定位基对象时进行反向引用，但对基对象的子条目则直接返回；“search”在定位基对象时不执行反向引用，但对基对象的子条目进行反向引用；“never”从不进行反向引用，直接返回条目本身；“always”总是执行反向引用
-A	只检索属性的名称，而不检索属性的值
-b basedn	指定用作搜索起始点的基对象的 DN。使用引号来指定该值，例如：“ou=West, o=Acme, c=US” 如果要搜索的服务器需要指定搜索起点，则必须使用此参数。否则此参数是可选的 也可以同时使用 -b 和 -s 来确定搜索范围。没有 -s，-b 就会搜索指定为起始点的项以及该项的所有子项
-c	遇到错误跳过，并继续执行
-f file	从文件中读取操作参数
-l limit	指定完成搜索的时间限制（秒）。如果没有指定此参数或指定的限制为 0，那么搜索就没有时间限制。但是，ldapsearch 的等待时间不会超过服务器上设置的搜索时间限制
-L	以 LDIFv1 格式输出响应
-LL	以 LDIF 格式输出响应，但不包含注释
-LLL	以 LDIF 格式输出响应，但包含注释和版本号
-P version	协议版本号，默认为 v3
-s scope	指定使用 -b 参数时的搜索范围，-b 和 -s 的参数出现的顺序并不重要 base—仅搜索 -b 参数指定的项 one—仅搜索 -b 参数指定项的直接子项，而不搜索该项本身 sub—搜索 -b 参数指定的项以及它的所有子项。这是不带 -s 时使用 -b 的默认行为
-S attr	按指定的属性排序结果
-t	把二进制值写到临时目录下的文件中
-tt	把所有值写到临时目录下的文件中
-T path	写文件到 path 路径下，而不是临时目录
-u	输出对用户友好的显示名
-z limit	指定返回项的最大数目。如果没有指定此参数或指定的限制为 0，那么返回的项没有数量限制。但是，ldapsearch 返回的项绝不会多于服务器允许的数量

表 6-9 搜索过滤条件

运 算 符	用 途	样 例
=	查找所包含的属性值与指定值相同的项	“cn=John Browning”
= <string>* <string>	查找所包含的属性值与指定的子字符串相同的项	“cn=John*” “cn=J*Brown”
>=	查找特定项，该项中包含的属性的数字或字母值大于或等于指定的值	“cn>=D”
<=	查找特定项，该项中包含的属性的数字或字母值小于或等于指定的值	“roomNumber<=300”
=*	查找包含特定属性的值的项，而不用管属性的值是什么	“sn=*”
~=	查找特定项，该项中所含属性的值约等于指定的值	“sn~=Brning” 可能返回 sn=Browning
&	查找与所有搜索过滤器中指定的条件相匹配的项	“(&(cn=John Browning)(l=Dallas))”
	查找与至少一个搜索过滤器中指定的条件相匹配的项	“((cn=John Browning)(l=Dallas))”
!	查找与任何搜索过滤器中指定的条件都不匹配的项	“(!(cn=John Browning)(l=Dallas))”

表 6-10 在查询过滤条件中必须进行转义的字符

字 符	十 进 制 值	十六进制值	转 义 串
*	42	0x2A	\2A
(40	0x28	\28
)	41	0x29	\29
\	92	0x5c	\5C
NUL (null 字节)	0	0x00	\00

为了更好地说明 `ldapsearch` 命令的使用方法，下面给出一些实例。

例子 1：查找 `ldap.example.com` 上的所有目录项，并返回所有属性和值。

```
ldapsearch -h ldap.example.com "objectClass=*"
```

例子 2：查找 `ldap.example.com` 上的所有项，并返回 `mail`、`cn`、`sn` 等属性。

```
ldapsearch -h ldap.example.com "objectClass=*" mail cn sn
```

例子 3：查找 `ldap.example.com` 上的“`ou=West, o=Acme, c=US`”条目下 `cn` 值以 `Mike` 开始的所有条目，并返回所有属性和值。

```
ldapsearch -b "ou=West, o=Acme, c=US" -h ldap.example.com "(cn=Mike*)"
```

例子 4：查找 `ldap.example.com` 上的所有条目并返回所有的属性和值，搜索时间限制为 5 秒。

```
ldapsearch -l 5 -h ldap.example.com "objectClass=*"
```

例子 5：返回 `ldap.example.com` 上的单条条目，把搜索基对象设为要返回条目的 DN，然后使用搜索范围为 `base` 的参数。

```
ldapsearch -h ldap.example.com -s base -b "uid=bjensen, ou=people, dc=example, dc=com" "(objectclass=*)"
```

例子 6：绑定用户“`cn=John Doe, ou=people, dc=example, dc=com`”，口令为明文“`password`”（此用户有查询权限），查找 `ldap.example.com` 上的所有条目，并以 LDIF 格式返回所有属性和值。

```
ldapsearch -h ldap.example.com -D "cn=john doe, ou=people, dc=example, dc=com" -w password -L "objectClass=*"
```

例子 7：组合过滤条件查询，如查找 `ldap.example.com` 上的属性为 `chaoyang` 或 `haidian` 的条目，返回所有属性。

```
ldapsearch -h ldap.example.com -s sub -b "dc=example, dc=com" "(&(|(L=chaoyang)(L=haidian)))"
```

6.4.2 应用接口编程与实例

《RFC 1823 LDAP 应用程序接口》定义了 LDAP 的 C API 接口。下文根据 RFC1823 进行说明，其他语言的 API 是相似的。

应用程序通常按以下 4 个步骤使用 LDAP API。

① 打开一个到 LDAP 服务器的连接，`ldap_open` 返回连接句柄，允许多个连接同时打开。

② LDAP 服务器验证用户身份。ldap_bind 及其他相关函数支持不同的认证方法。

③ 执行 LDAP 操作获取结果。ldap_search 及相关函数返回的结果可以由 ldap_result2error、ldap_first_entry、ldap_next_entry 解析。

④ 关闭连接。调用 ldap_unbind 实现。

操作能够同步或异步执行。同步调用以_s 结尾，所以同步搜索的函数为 ldap_search_s。所有同步程序返回一个代表操作结果的指示符。（例如，常量 LDAP_SUCCESS 或其他错误码。）异步程序返回操作初始化的消息 ID，此 ID 可以被随后的 ldap_result 调用取得操作结果集。一个异步操作可以被 ldap_abandon 函数取消。

1. 调用 LDAP 操作

所有调用使用一个“连接句柄”（一个指向 LDAP 结构的指针），此结构包含每一个连接的信息，许多函数都返回 LDAPMessage 结构。

(1) 打开一个连接

ldap_open 打开一个到 LDAP 服务器的连接。

```
typedef struct ldap {
    /* ... opaque parameters ... */
    int      ld_deref;
    int      ld_timelimit;
    int      ld_sizelimit;
    int      ld_errno;
    char     *ld_matched;
    char     *ld_error;
    /* ... opaque parameters ... */
} LDAP;

LDAP* ldap_open(char* hostname, int portno);
```

参数：

hostname：要连接的运行 LDAP 服务器地址，以空格分隔的主机名或 IP 地址列表。客户端依次尝试连接这些主机名，直到有一个连接成功。

portno：要连接的 TCP 端口号。

成功时，ldap_open 返回一个“连接句柄”，如果不能打开连接则返回 NULL。

(2) 目录验证

ldap_bind 和相关函数用来进行目录验证。

```
int ldap_simple_bind( LDAP *ld, char *dn, char *passwd );
int ldap_simple_bind_s( LDAP *ld, char *dn, char *passwd );
int ldap_sasl_bind(LDAP *ld, const char *dn, const char *mechanism,
    const struct berval *cred, LDAPControl **serverctrls,
    LDAPControl **clientctrls, int *msgidp);
int ldap_sasl_bind_s(LDAP *ld, const char *dn, const char *mechanism,
    const struct berval *cred, LDAPControl **serverctrls,
    LDAPControl **clientctrls, struct berval **servercredp);
```

参数:

ld: 连接句柄。

dn: 绑定的条目名称。

passwd: ldap_simple_bind 使用的密码, 如果为 NULL, 则传递长度为 0 的密码到服务端。

mechanism: 传递 LDAP_SASL_SIMPLE_NULL 以使用简单认证, 或标识 SASL 方法的字符串。

cred: 被验证的身份, mechanism 会确定 cred 的数据格式。

serverctrls: 服务器端控制列表, 如果为 NULL, 表示没有服务器端控制。

clientctrls: 客户器端控制列表, 如果为 NULL, 表示没有使用客户端控制。

msgidp: 存储消息标识, 用于获取操作结果。

servercredp: 返回的服务器端身份标识、出错或无服务器端标识, 设置为 NULL。

成功时, 返回 LDAP_SUCCESS。

(3) 关闭连接

ldap_unbind 用来与一个目录解除绑定并关闭连接。

```
Int ldap_unbind(LDAP* ld);
```

参数:

ld: 连接句柄。

ldap_unbind 同步状态工作, 同目录解除绑定, 关闭连接, 在返回前释放 ld 结构空间。

ldap_unbind 返回 LDAP_SUCCESS (或其他请求不能送到 LDAP Server 的 LDAP 错误码)。

调用 ldap_unbind 后, ld 连接句柄将不可用。

(4) 查询

ldap_search 及其相关函数用来对 LDAP 目录进行查询, 返回请求的每一个匹配条目的属性集。下面是三个相关函数:

```
struct timeval {
    long    tv_sec;
    long    tv_usec;
};

int ldap_search( LDAP *ld, char *base, int  scope, char  *filter, char *attrs[], int attrsonly );
int ldap_search_s( LDAP *ld, char *base, int  scope, char *filter, char *attrs[], int  attrsonly,
                  LDAPMessage **res );
int ldap_search_st( LDAP *ld, char *base, int  scope, char *filter, char *attrs[], int  attrsonly, struct
                  timeval *timeout, LDAPMessage **res );
```

参数:

ld: 连接句柄。ld 连接句柄的 3 个字段控制查询如何执行, 它们是 ld_sizelimit、ld_timelimit、ld_deref。ld_sizelimit: 限制查询返回的条目数量, 0 代表无限制。ld_timelimit: 限制查询时间, 以秒为单位, 0 代表无限制。ld_deref: 常量 LDAP_DERF_NEVER、LDAP_DEREF_SEARCHING、LDAP_DEREF_FINDING、LDAP_DEREF_ALWAYS 之一, 描述了在查询过程中如何处理别名。LDAP_DEREF_SEARCHING 意为在查询中别名被解除, 但不会在定位于查询的基对象时解除引用。LDAP_DERED_FINGINFG 意为别名在定位基对

象但不是在查询过程中解除引用。异步查询由 `ldap_search` 初始化。函数返回查询初始化消息 ID。查询结果将被随后的 `ldap_result` 调用取得。结果由结果解释程序解释，在下面将详细介绍。如果出错，返回-1，同时 `ld` 连接句柄的 `ld_errno` 字段将会被相应设置。

base: 开始搜索的 dn 条目。

scope: 常量 `LDAP_SCOPE_BASE`、`LDAP_SCOPE_ONELEVEL`、`LDAP_SCOPE_SUBTREE` 之一，表示搜寻范围。

filter: RFC 1558 定义的字符串，表示搜索条件。

attr: 指明要返回的属性，NULL 串将返回所有可用的属性。

attronly: 布尔值，为 0 时指明返回属性的类型和属性值，非 0 值只返回所需类型。

timeout: 在调用 `ldap_search_st` 时，定义本地查询的超时时间。

res: 在同步调用时使用，作为返回结果参数，包含查询调用完成的结果。

同步查询通过调用 `ldap_search_s` 和 `ldap_search_st` 执行。这些函数除了 `ldap_search_st` 多一个附加参数以定义查询超时外，都是相同的。两个函数返回查询结果的标识、`LDAP_SUCCESS` 或者错误标识。查询条目返回结果包含在 `res` 参数中。此参数对于调用者是非透明的，条目、属性、值等应被后面说明的函数解析出来，包含在 `res` 中的结果应调用 `ldap_msgfree` 释放空间。

(5) 读一个条目

LDAP 不支持直接的读操作，代之以查询来模拟，设置 `base` 参数为要读的条目 DN，`scope` 设置为 `LDAP_SCOPE_BASE`，`filter` 设置为 “(objectclass=*)”。`attrs` 包含返回的属性列表。

(6) 列出一个条目的子条目

LDAP 不支持直接列表操作，代之以查询来模拟，设置 `base` 为要列表的条目 DN，`scope` 设为 `LDAP_SCOPE_ONELEVEL`，`filter` 设为 “(objectclass=*)”。`attrs` 包含返回的每一个子条目的属性列表。

(7) 修改条目

`ldap_modify` 和 `ldap_modify_s` 函数用来修改已存在的 LDAP 条目。

```
typedef struct ldapmod {
    int    mod_op;
    char *mod_type;
    union {
        char **modv_strvals;
        struct berval **modv_bvals;
    } mod_vals;
} LDAPMod;

#define mod_values      mod_vals.modv_strvals
#define mod_bvalues     mod_vals.modv_bvals

int ldap_modify( LDAP *ld, char *dn, LDAPMod *mods[] );
int ldap_modify_s( LDAP *ld, char *dn, LDAPMod *mods[] );
```

参数:

ld: 连接句柄。

dn: 要修改的条目名。

mods: 设置修改条目的空结尾的数组。此参数在 LDAPMod 结构中有如下意义。

mod_op: 执行的修改操作, 为 LDAP_MOD_ADD、LDAP_MOD_DELETE、LDAP_MOD_REPLACE 之一。此字段也指明在 mod_vals 联合中的值的类型。此字段同 LDAP_MOD_BVALUES 进行或操作, 以指定 mod_bvalues 值的形式。另外, 也可以使用 mod_values 形式。

mod_type: 要修改的类型。

mod_vals: 对此值进行 add、delete、replace 操作。仅 mod_values 或 mod_bvalues 之一可以使用。mod_values 为以 NULL 结尾的字符串数组, mod_bvalues 为以 NULL 结尾的 berval 结构数组, 可以用来传送图像一类的二进制值。

对于 LDAP_MOD_ADD 操作, 给定的值是要添加的条目, 根据需要创建属性。

对于 LDAP_MOD_DELETE 操作, 给定的值要从条目中删除, 如果条目属性被删除, mod_vals 字段应设为 NULL。

对于 LDAP_REPLACE 操作, 对属性进行替换, 或根据需要创建。所有的修改操作按列出的顺序执行。

ldap_modify_s 根据修改操作结果返回 LDAP 错误码。此代码由 ldap_error 及相关函数解释。ldap_modify 返回请求初始化消息 ID, 或出错时返回-1。操作结果由 ldap_result 获得。

(8) 修改条目的 RDN

ldap_modrdn 和 ldap_modrn_s 函数用来修改 LDAP 条目名。

```
int ldap_modrdn( LDAP *ld, char *dn, char *newrdn, int deleteoldrdn);
int ldap_modrdn_s(LDAP *ld, char *dn, char *newrdn, int deleteoldrdn );
```

参数:

ld: 连接句柄。

dn: 要修改的条目名。

newrdn: 条目的新的 RDN。

deleteoldrdn: 布尔值, 非 0 值指明删除旧的 RDN, 0 值指明旧的 RDN 应保留为条目的非区别值。

ldap_modrdn_s 函数为同步调用, 返回指明操作输出的 LDAP 错误码。ldap_modrdn 函数为异步调用, 返回初始化操作的消息 ID, 或者出错返回-1。由 ldap_result 取得返回结果。

(9) 添加条目

ldap_add 和 ldap_add_s 用来添加 LDAP 目录条目。

```
int ldap_add( LDAP *ld, char *dn, LDAPMod *attrs[] );
int ldap_add_s( LDAP *ld, char *dn, LDAPMod *attrs[] );
```

参数:

ld: 连接句柄。

dn: 要添加的条目名。

attrs: 条目属性, 使用为 ldap_modify 定义的 LDAPMod 结构。mod_type 和 mod_vals 字段需要填充, mod_op 字段会被忽略, 除非同 LDAP_MOD_BVALUES 进行逻辑或运算 (|) 后, 用来指定值为 mod_vals 中的 mod_bvalues。

注意: 条目的父条目必须存在。

ldap_add_s 为同步函数, 返回指明操作输出的 LDAP 错误码。ldap_add 为异步函数, 返回操作初始化消息 ID, 或出错时返回-1, 结果由 ldap_result 函数取得。

(10) 删除条目

ldap_delete 和 ldap_delete_s 用来从 LDAP 目录中删除条目。

```
int ldap_delete(LDAP* ld, char* dn);
int ldap_delete_s(LDAP* ld, char* dn);
```

参数:

ld: 连接句柄。

dn: 要删除的条目名。

注意: 要删除的条目必须为叶子条目 (例如, 它不能有子条目)。LDAP 不支持删除条目子树的操作。

ldap_delete_s 为同步函数, 返回指明操作输出的 LDAP 错误代码。ldap_delete 为异步函数, 返回操作初始化消息 ID, 或出错时返回-1, 结果由 ldap_result 函数取得。

2. 操作放弃

ldap_abandon 用来放弃一个操作过程。

```
int ldap_abandon(LDAP* ld, int msgid);
```

ldap_abandon 放弃消息 ID 为 msgid 的操作。如果操作成功返回 0, 否则返回-1。成功调用 ldap_abandon 后, 给定消息 ID 的结果不会由 ldap_result 调用返回。

3. 取得结果的调用

ldap_result 用来取得先前同步初始化的结果。ldap_msgfree 用来释放先前调用 ldap_result 或同步查询函数取得的结果。

```
int ldap_result(LDAP *ld, int msgid, int all, struct timeval *timeout, LDAPMessage **res);
int ldap_msgfree( LDAPMessage *res );
```

参数:

ld: 连接句柄。

msgid: 需要返回结果的操作的消息 ID。

all: 布尔值, 代表查询结果的含义, 非 0 值指明在所有查询结果都应取得后才能返回。如为 0, 查询结果 (条目) 将会一次返回查到的一个。

timeout: 表示等待返回结果的超时时间。NULL 值将造成 ldap_result 阻塞等待, 直到结果可用。timeout 值为 0 秒表示轮询状态。

res: 对于 ldap_result, 是一个包含操作结果集的结果参数。对于 ldap_msgfree, 表示要被释放的结果参数从先前的 ldap_result、ldap_search_s 或 ldap_seatch_st 调用取得。

在成功完成后, ldap_result 返回结果的类型, 这些类型为以下常量: LDAP_RES_BIND, LDAP_RES_SEARCH_ENTRY, LDAP_RES_SEARCH_RESULT, LDAP_RES_MODIFY, LDAP_RES_ADD, LDAP_RES_DELETE, LDAP_RES_MODRDN, LDAP_RES_COMPARE。

ldap_result 在超时后返回 0, 出错后返回-1。在这种情况下, ld 结构的 ld_err 字段会相应设置。

ldap_msgfree 释放指向 res 的结果结构并返回释放的消息类型。

4. 错误处理调用

下面的调用捕获其他 LDAP API 返回的错误。

```
int ldap_result2error( LDAP *ld, LDAPMessage *res, int freeit );
char* ldap_err2string(int err);
void ldap_perror(LDAP* ld, char* msg);
```

参数:

ld: 连接句柄。

res: ldap_result 返回的 LDAP 操作结果, 或一个异步 API 操作调用的结果。

freeit: 布尔值, 指明 res 参数是否被释放 (非 0 值释放, 0 值不释放)。

err: LDAP 错误码。

msg: 在 LDAP 错误信息之前显示的信息。

ldap_result2error 用来将 res 参数转换成数字形式的错误码。同时也将结果信息中的 ld_matched 和 ld_error 部分解析出来, 并将它们放入连接句柄信息中。所有的同步操作函数在返回前调用 ldap_result2error, 确保这些字段正确设置。

连接结构的相关字段如下。

ld_matched: 此参数包含 LDAP 服务器返回结果的错误信息。

ld_errno: 指明操作 LDAP 错误码。

ldap_err2string 用来将 LDAP 错误码转换成以 NULL 结尾的包含错误描述的字符串。此错误码为 ldap_result2error 或一个同步 API 操作调用的返回结果。

5. 解释查询条目的调用

以下函数用来解析 ldap_search 及相关函数返回的条目。这些条目返回一个只应被下面函数访问的非透明的结构。这些函数提供遍历所返回的条目, 遍历条目的属性, 取得条目名称, 取得条目的给定属性的属性值。

(1) 遍历条目集

ldap_first_entry 和 ldap_next_entry 函数用来遍历查询结果的条目集。

ldap_count_entries 用来计算返回的条目数量。

```
LDAPMessage* ldap_first_entry(LDAP* ld, LDAPMessage* res);
LDAPMessage* ldap_next_entry(LDAP* ld, LDAPMessage* entry);
```

```
int ldap_count_entry(LDAP* ld, LDAPMessage* res);
```

参数:

ld: 连接句柄。

res: 查询结果。

entry: 先前的 ldap_first_entry 或 ldap_next_entry 调用返回的条目。

ldap_first_entry 和 ldap_next_entry 在结果中没有条目时返回 NULL。当在遍历条目时发生错误也返回 NULL, 这种情况下, ld 连接句柄的 ld_errno 字段会被设置为错误码。

ldap_count_entries 返回条目链的条目数, 它也用来计算 ldap_first_entry 或 ldap_next_entry 返回的条目链中的剩余条目数。

(2) 遍历条目属性

ldap_first_attribute 和 ldap_next_attribute 调用用来遍历一个条目的属性列表。

```
char *ldap_first_attribute(LDAP *ld, LDAPMessage *entry, void **ptr);
char *ldap_next_attribute(LDAP *ld, LDAPMessage *entry, void *ptr);
```

参数:

ld: 连接句柄。

entry: 要遍历的条目, 由 ldap_first_entry 或 ldap_next_entry 返回。

ptr: 在 ldap_first_attribute 中, 用来内部跟踪当前条目位置的指针地址, 在 ldap_next_attribute 中, 由先前的 ldap_first_attribute 调用返回的指针。

ldap_first_attribute 和 ldap_next_attribute 在到达属性结尾时会返回 NULL, 或者如果发生错误, ld 连接句柄的 ld_errno 字段会被设置为错误码。

两个函数返回一个指向预先连接的包含当前属性名的缓冲区, 这应看作静态数据。ldap_first_attribute 将会分配空间并返回指向用来跟踪当前位置的 BerElement 类型指针。此指针应传给后面调用的 ldap_next_attribute 来遍历条目属性。

返回的属性名由 ldap_get_values 及相关函数解析并取得相关联的值。

(3) 获取属性值

ldap_get_values 和 ldap_get_values_len 用来从条目中取得给定属性的值。ldap_count_values 和 ldap_count_values_len 用来计算返回值的个数。ldap_value_free 和 ldap_value_free_len 用来释放属性值。

```
typedef struct berval {
    unsigned long   bv_len;
    char *bv_val;
};

char **ldap_get_values(LDAP *ld, LDAPMessage *entry, char *attr);
struct berval **ldap_get_values_len(LDAP *ld, LDAPMessage *entry, char *attr);
int ldap_count_values( char **vals );
int ldap_count_values_len( struct berval **vals );
int ldap_value_free( char **vals );
int ldap_value_free_len( struct berval **vals );
```

参数:

ld: 连接句柄。

entry: 获取属性值的条目。

attr: 需要获取的属性值。

vals: 调用 ldap_get_values 或 ldap_get_values_len 返回的值。

上面提供了两种形式的变量调用, 第一种形式仅适于非二进制的字符串数据, 第二种以 len 结尾的形式可以使用任何类型的数据。

注意, 返回内存在不使用时, 为了防止内存泄露, 不要忘记调用 ldap_value_free 或 ldap_value_free_len 释放。

(4) 取得条目名称

ldap_get_dn 用来返回条目名称。

ldap_explode_dn 用来拆分名称为部分名。

ldap_dn2ufn 用来转换名称为更多的“用户友好”的格式。

```
char *ldap_get_dn( LDAP *ld, LDAPMessage *entry );
char **ldap_explode_dn( char *dn, int notypes );
char *ldap_dn2ufn( char *dn );
```

参数:

ld: 连接句柄。

entry: 需要取得名称的条目。

dn: 要拆分的 dn, 由 ldap_get_dn 返回。

notypes: 布尔型参数, 如果非 0, 则 dn 的各部分应该取消类型名称。(例如, “cn=Babs” 应变成 “Babs”。)

ldap_get_dn 在解析 dn 发生错误时返回 NULL, 设置 ld 连接句柄的 ld_errno 字段以指明错误。返回的指针由函数内部申请空间, 所以在不用时应调用 ldap_memfree 释放。

ldap_explode_dn 返回包含 DN 的 RDN 部分字符串数组, 带有或不带类型由 notypes 参数指定。返回的数组在不再使用时应调用 ldap_value_free 释放。

ldap_dn2ufn 转换 DN 为用户友好的格式(参照 RFC 1781)。UFN 返回的是已分配的空间, 应在不用时调用 ldap_memfree 释放。

6. 简单 LDAP API 代码

```
#include <ldap.h>
int main(int argc, char *argv[])
{
    LDAP *ld;
    LDAPMessage *res, *e;
    int i;
    char *a, *dn;
    void *ptr;
    char **vals;
```

```

/* 打开连接 */
ld = ldap_open("dotted.host.name", LDAP_PORT);
if (ld == NULL) exit( 1 );

/* 以 nobody 身份进行认证 */
if (ldap_simple_bind_s(ld, NULL, NULL) != LDAP_SUCCESS)
{
    ldap_perror(ld, "ldap_simple_bind_s");
    exit(1);
}

/* 搜索 cn 为"Babs Jensen"的条目，返回所有属性 */
if (ldap_search_s(ld, "o=University of Michigan, c=US",
    LDAP_SCOPE_SUBTREE, "(cn=Babs Jensen)", NULL, 0, &res)
    != LDAP_SUCCESS)
{
    ldap_perror( ld, "ldap_search_s" );
    exit( 1 );
}

/* step through each entry returned */
for ( e = ldap_first_entry( ld, res ); e != NULL; e = ldap_next_entry( ld, e ) )
{
    /* print its name */
    dn = ldap_get_dn( ld, e );
    printf( "dn: %s0, dn );
    Ldap_memfree(dn);

    /* 输出每个属性 */
    for ( a = ldap_first_attribute( ld, e, &ptr ); a != NULL;
        a = ldap_next_attribute( ld, e, ptr ) )
    {
        printf( "attribute: %s0, a );
        /* 输出每个属性值 */
        vals = ldap_get_values( ld, e, a );
        for ( i = 0; vals[i] != NULL; i++ )
        {
            printf( "value: %s0, vals[i] );
        }
        ldap_value_free( vals );
    }
}

```

```

/* 释放搜索结果集 */
ldap_msgfree( res );

/* 关闭和释放连接资源 */
ldap_unbind( ld );
}

```

6.4.3 LDAP 应用案例

LDAP 的优势是查询速度快，以信息树的形式存储数据，LDAP 特别适合身份认证、资源管理、CA 等系统的数据管理和发布，这些系统的数据符合目录信息树的特点。例如，身份认证系统管理的是一个机构的实体信息，而机构组织结构本身就可以表示成一个信息树。在 CA 系统中，证书的颁发者和使用者就是使用与 LDAP 完全等价的 DN 表示的，用 LDAP 存储要发布的证书和 CRL 就非常合适。在 CA 系统中一般采用如图 6-9 所示的逻辑结构。

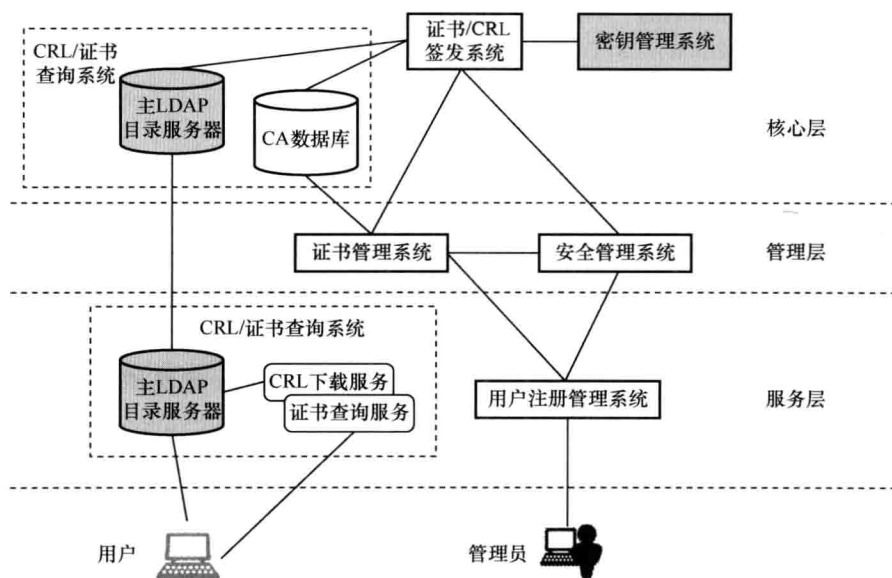


图 6-9 CA 系统逻辑结构

从图中可以看出 CA 系统分三层结构，分别是核心层、管理层、服务层，其中核心层的安全级别最高，主目录服务器部署在核心层，由“证书/CRL 签发系统”向主目录写入已签发的证书和 CRL。而向用户提供的证书和 CRL 查询服务由从目录服务提供，从目录服务实时从主目录服务同步证书和 CRL。

第7章 实 验 一

7.1 DER 编码示例: X.501 Name 类型

7.1.1 ASN.1 描述与实例

1. ASN.1 描述

X.501 Name 类型用 ASN.1 描述如下:

```
Name ::= CHOICE { RDNSequence }
RDNSequence ::= SEQUENCE OF RelativeDistinguishedName
RelativeDistinguishedName ::= SET OF AttributeValueAssertion
AttributeValueAssertion ::= SEQUENCE {
    AttributeType,
    AttributeValue }
AttributeType ::= OBJECT IDENTIFIER
AttributeValue ::= ANY
```

Name 类型定义为 CHOICE 类型, 目前只有 1 个选项 RDNSequence。RDNSequence 定义为 SEQUENCE OF 类型, 由 0 个或多个 RelativeDistinguishedName 组成。RelativeDistinguishedName 定义为 SET OF 类型, 由 0 个或多个 AttributeValueAssertion 组成。AttributeValueAssertion 定义为 SEQUENCE 类型, 由 2 个成分组成: 1 个为 AttributeType 类型和 1 个 AttributeValue 类型。AttributeType 定义为 OBJECT IDENTIFIER 类型。AttributeValue 定义为 ANY 类型, 具体内容 by AttributeType 决定。

事实上, Name 类型可理解为分层或树形结构, 即 X.500 目录树结构。

2. Name 实例

对于用户 Test User 1, 其对应的 Name 类型采用分层结构描述为:

```
(root)
|
countryName = "US"
|
organizationName = "Example Organization"
|
commonName = "Test User 1"
```

其中, 每层对应一个 RelativeDistinguishedName; 每个 RelativeDistinguishedName 由 1 个 AttributeValueAssertion 组成。等号前内容为 AttributeType, 等号后内容为 AttributeValue。

用户 Test User 1 包含 3 个 AttributeType: countryName、organizationName、commonName, 其 OID 定义如下:

```

attributeType OBJECT IDENTIFIER ::= { joint-iso-ccitt(2) ds(5) 4 }
countryName OBJECT IDENTIFIER ::= { attributeType 6 }
organizationName OBJECT IDENTIFIER ::= { attributeType 10 }
commonName OBJECT IDENTIFIER ::= { attributeType 3 }

```

7.1.2 DER 编码过程

1. AttributeType 编码

AttributeType 为 OBJECT IDENTIFIER 基本类型，编码规则采用基本类型定长模式。

对于标识串，采用低标识编码方式，只需 1 个字节。OBJECT IDENTIFIER 的 tag 为 0x06；class 选择 universal，则位 8 和位 7 为 0，OBJECT IDENTIFIER 为基本类型，则位 6 为 0。因此，标识串=0x06。

对于长度串，采用短型编码方式，只需 1 个字节。

对于内容串，由 3 个字节组成。2.5.4.6 编码为 55 04 06，2.5.4.10 编码为 55 04 0A，2.5.4.3 编码为 55 04 03。

具体编码过程如表 7-1 所示。

表 7-1 AttributeType 编码过程

AttributeType	OID 定义	标 识 串	长 度 串	内 容 串
countryName	2.5.4.6	06	03	55 04 06
organizationName	2.5.4.10	06	03	55 04 0A
commonName	2.5.4.3	06	03	55 04 03

2. AttributeValue 编码

AttributeValue 为 PrintableString 基本类型，编码规则采用基本类型定长模式。

对于标识串，采用低标识编码方式，只需 1 个字节。PrintableString 的 tag 为 0x13；class 选择 universal，则位 8 和位 7 为 0，OBJECT IDENTIFIER 为基本类型，则位 6 为 0。因此，标识串=0x13。

对于长度串，采用短型编码方式，只需 1 个字节。

对于内容串，由其 ASCII 码组成。

具体编码过程如表 7-2 所示。

表 7-2 AttributeValue 编码过程

AttributeValue	标识串	长度串	内容串
"US"	13	02	55 53
"Example rganization"	13	14	45 78 61 6D 70 6C 65 20 4F 72 67 61 6E 69 7A 61 74 69 6F 6E
"Test User 1"	13	0B	54 65 73 74 20 55 73 65 72 20 31

3. AttributeValueAssertion 编码

AttributeValueAssertion 为 SEQUENCE 结构类型，编码规则采用结构类型定长模式。

对于标识串，采用低标识编码方式，只需 1 个字节。SEQUENCE 的 tag 为 0x10；class

选择 universal, 则位 8 和位 7 为 0, SEQUENCE 为结构类型, 则位 6 为 1。因此, 标识串=0x30。

对于长度串, 采用短型编码方式, 只需 1 个字节。

对于内容串, 由 AttributeType 和 AttributeValue 的 DER 编码值组成。

具体编码过程如表 7-3 所示。

表 7-3 AttributeValueAssertion 编码过程

AttributeValueAssertion	标 识 串	长 度 串	内 容 串
countryName ="US"	30	09	06 03 55 04 06 13 02 55 53
organizationName= "Example rganization"	30	1B	06 03 55 04 0A 13 14 45 78 ... 6F 6E
commonName = "Test User 1"	30	12	06 03 55 04 03 13 0B 54 65 ... 20 31

4. RelativeDistinguishedName 编码

RelativeDistinguishedName 为 SET OF 结构类型, 编码规则采用结构类型定长模式。

对于标识串, 采用低标识编码方式, 只需 1 个字节。SET OF 的 tag 为 0x11; class 选择 universal, 则位 8 和位 7 为 0, SET OF 为结构类型, 则位 6 为 1。因此, 标识串=0x31。

对于长度串, 采用短型编码方式, 只需 1 个字节。

对于内容串, 由 AttributeValueAssertion 的 DER 编码值组成。

具体编码过程如表 7-4 所示。

表 7-4 RelativeDistinguishedName 编码过程

RelativeDistinguishedName	标 识 串	长 度 串	内 容 串
countryName ="US"	31	0B	30 09 06 03 55 04 06 13 02 55 53
organizationName= "Example rganization"	31	1D	30 1B 06 03 55 04 0A 13 14 45 78 ... 6F 6E
commonName = "Test User 1"	31	14	30 12 06 03 55 04 03 13 0B 54 65 ... 20 31

5. RDNSequence 编码

RDNSequence 为 SEQUENCE OF 结构类型, 编码规则采用结构类型定长模式。

对于标识串, 采用低标识编码方式, 只需 1 个字节。SEQUENCE OF 的 tag 为 0x10; class 选择 universal, 则位 8 和位 7 为 0, SEQUENCE OF 为结构类型, 则位 6 为 1。因此, 标识串=0x30。

对于长度串, 采用短型编码方式, 只需 1 个字节。

对于内容串, 由 3 个 RelativeDistinguishedName 的 DER 编码值组成。

具体编码过程如表 7-5 所示。

表 7-5 RDNSequence 编码过程

RDNSequence	标 识 串	长 度 串	内 容 串
countryName ="US"	30	42	31 0B 30 09 06 03 55 04 06 13 02 55 53 31 1D 30 1B 06 03 55 04 0A 13 14 45 78 ... 6F 6E 31 14 30 12 06 03 55 04 03 13 0B 54 65 ... 20 31

6. Name 编码

Name 为 CHOICE 类型，其 DER 编码值与 RDNSequence 相同。

用户 Test User 1 最终 DER 编码值如表 7-6 所示。

表 7-6 用户 Test User 1 最终 DER 编码值

DER 编码值	ASN.1 描述
30 42 31 0B 30 09 06 03 55 04 06 13 02 55 53 31 1D 30 1B 06 03 55 04 0A 13 14 45 78 61 6D 70 6C 65 20 4F 72 67 67 61 6E 69 7A 61 74 69 6F 6E 31 14 30 12 06 03 55 04 03 13 0B 54 65 73 74 20 55 73 65 72 20 31	attributeType = countryName attributeValue = "US" attributeType = organizationName attributeValue = "Example Organization" attributeType = commonName attributeValue = "Test User 1"

7.2 RSA 算法示例

7.2.1 密钥产生

1. 计算 n

算法：选择两个大素数 p 和 q ，计算出 $n=pq$ 。

示例：选择 2 个素数： $p=7$ ， $q=17$ 。 $n=pq=7\times 17=119$ 。

2. 计算 $\varphi(n)$

算法：计算出 n 的欧拉函数 $\varphi(n)=(p-1)(q-1)$ 。

示例： $\varphi(n)=(7-1)\times(17-1)=96$ 。

3. 选择 e

算法：从 $[0, \varphi(n)-1]$ 中选择一个与 $\varphi(n)$ 互素的数 e 作为公开的加密指数。

示例：从 $[0, 95]$ 中选择一个与 96 互素的数 $e=5$ 。

4. 计算 d

算法：计算出满足公式 $ed=1 \bmod \varphi(n)$ 的 d 作为解密指数。

示例：根据 $5d=1 \bmod 96$ ，得出 $d=77$ 。（ $ed=5\times 77=385=4\times 96+1=1 \bmod 96$ ）

5. 获得公钥和私钥

算法：公钥 $PK=\{e, n\}$ ，私钥 $SK=\{d, n\}$

示例：公钥 $PK=\{5, 119\}$ ，私钥 $SK=\{77, 119\}$

7.2.2 加密解密

若用整数 X 表示明文，用整数 Y 表示密文（ X 和 Y 均小于 n ）。

1. 公钥加密

算法： $Y=X^e \bmod n$ ， $PK=\{e, n\}$

示例：设 $X=19$ ， $PK=\{5, 119\}$

$Y=19^5 \bmod 119 = 2476099 \bmod 119 = (119 \times 20807 + 66) \bmod 119 = 66$ 。

即：明文 $X=19$ ，密文 $Y=66$ 。

2. 私钥解密

算法： $X=Y^d \bmod n$ ， $SK=\{d, n\}$

示例：设 $Y=66$ ， $SK=\{77, 119\}$

$X=66^{77} \bmod 119 = 1.27 \cdots \times 10^{140} \bmod 119 = (1.06 \cdots \times 10^{138} + 19) \bmod 119 = 19$ 。

即：密文 $Y=66$ ，明文 $X=19$ 。

第三部分

PKI 之数字证书与私钥： 网络身份证

第 8 章 公/私钥格式

8.1 RSA

PKCS #1 中规定了 RSA 公钥/私钥的具体格式。

1. RSA 公钥格式

RSA 公钥格式用 ASN.1 描述如下：

```
RSAPublicKey ::= SEQUENCE {  
    modulus          INTEGER, -- n  
    publicExponent   INTEGER  -- e  
}
```

其中，n 和 e 为 RSA 公钥参数。

2. RSA 私钥格式

RSA 私钥格式用 ASN.1 描述如下：

```
RSAPrivateKey ::= SEQUENCE {  
    version          Version,  
    modulus          INTEGER, -- n  
    publicExponent   INTEGER, -- e  
    privateExponent  INTEGER, -- d  
    prime1           INTEGER, -- p  
    prime2           INTEGER, -- q  
    exponent1        INTEGER, -- d mod (p-1)  
    exponent2        INTEGER, -- d mod (q-1)  
    coefficient       INTEGER, -- (inverse of q) mod p  
}  
version ::= INTEGER
```

其中，version 用于区分格式版本，缺省值为 0。

3. RSA 加密格式

RSA 公钥和私钥均可进行加密和解密操作。

假设使用公钥 pk (RSAPublicKey 类型) 或私钥 vk (RSAPrivateKey 类型) 对明文数据 D (字符串类型) 进行加密计算。具体计算步骤如下：

① 构造加密块 (encryption block)：EB = 00 || BT || PS || 00 || D。EB 长度为 k。

其中，BT 为块类型，OCTET STRING 类型，长度=1，值可以为 00、01 或 02。私钥加密/解密时，BT=00 或 01，公钥加密/解密时，BT=02。

PS 为填充字符串, 为 OCTET STRING 类型, 长度 = $k - 3 - \|D\|$ 。当 BT=00 时, PS 值全为 00。当 BT=01 时, PS 值全为 FF。当 BT=02 时, PS 值不能为 0, 随机产生。

② 格式转换: 将 OCTET STRING 类型 EB 转换成 Integer 类型 x 。

③ RSA 加密: 使用 RSA 公钥或私钥对 Integer 类型 x 进行加密后获得 Integer 类型密文 y 。

④ 格式转换: 将 Integer 类型密文 y 转换成 OCTET STRING 类型密文 ED, 长度为 k 。

假设使用公钥 pk (RSAPublicKey 类型) 或私钥 vk (RSAPrivateKey 类型) 对密文数据 ED (OCTET STRING 类型) 进行解密计算。具体计算步骤如下:

① 将 OCTET STRING 类型 ED 转换成 Integer 类型密文 y 。

② RSA 解密: 使用 RSA 公钥或私钥对 Integer 类型密文 y 进行解密后获得 Integer 类型明文 x 。

③ 格式转换: 将 Integer 类型明文 x 转换成 OCTET STRING 类型明文 EB, 长度为 k 。

④ 加密块分拆 (encryption block): 将明文 EB 分拆成 BT、PS 和 D, 即可获得明文数据 D。

4. RSA 签名格式

RSA 签名时摘要格式用 ASN.1 描述如下:

```
DigestInfo ::= SEQUENCE {
    digestAlgorithm DigestAlgorithmIdentifier,
    digest Digest }
DigestAlgorithmIdentifier ::= AlgorithmIdentifier
Digest ::= OCTET STRING
AlgorithmIdentifier ::= SEQUENCE {
    Algorithm          OBJECT IDENTIFIER,
    Parameters         ANY DEFINED BY algorithm OPTIONAL }
```

其中, DigestAlgorithm 为摘要算法, Digest 为摘要值。

假设使用私钥 vk (RSAPrivateKey 类型) 对待签名数据 M (字符串类型) 进行签名计算。具体计算步骤如下:

① 计算摘要值: $MD = \text{HASH}(M)$ 。

② 数据编码: 将摘要算法 ID 和摘要值 MD 按照 DigestInfo 类型进行编码, 获得摘要数据 D。

③ RSA 加密: 使用私钥 vk 对摘要数据 D 进行加密后获得密文 ED (OCTET STRING 类型), 其中 BT=0x01。

④ 格式转换: 将密文 ED 转换成比特字符串 S (BIT STRING 类型)。其中 S 采用 MSB (most significant bit) 方式。S 即为签名结果。

假设使用公钥 pk (RSAPublicKey 类型) 对待签名数据 M (字符串类型) 和待签名结果 S (BIT STRING 类型) 进行签名验证。具体计算步骤如下:

① 格式转换: 将签名结果 S (BIT STRING 类型) 转换成密文 ED (OCTET STRING 类型)。

② RSA 解密：使用公钥 pk 对密文 ED 进行解密获得摘要数据 D（OCTET STRING 类型）。

③ 数据解码：摘要数据 D 是 DigestInfo 类型，解码后获得摘要值 MD 和摘要算法 ID。

④ 计算摘要并比较：根据摘要算法 ID，计算 M 的摘要值 MD'= HASH（M）。如果 MD' 等于 MD，则验证成功，否则验证失败。

8.2 SM2

《SM2 密码算法使用规范》规定了数字证书 SM2 算法的公/私钥及加密签名格式。

1. SM2 公钥格式

SM2 公钥格式用 ASN.1 描述如下：

SM2PublicKey ::= BIT STRING

SM2 公钥是 SM2 曲线上的一个点，由横坐标和纵坐标两个分量来表示，记为 (x, y)，简记为 Q，每个分量长度为 256 位。SM2PublicKey 内容格式为：04 || X || Y。其中，X 和 Y 分别表示公钥的 x 分量和 y 分量，其长度各为 256 位。

2. SM2 私钥格式

SM2 私钥格式用 ASN.1 描述如下：

SM2PrivateKey ::= INTEGER

SM2 私钥是一个大于或等于 1 且小于 n-1 的整数（n 为 SM2 算法的阶），简记为 k，长度为 256 位。

3. SM2 加密数据格式

SM2 加密数据格式用 ASN.1 描述如下：

```
SM2Cipher ::= SEQUENCE {
    XCoordinate    INTEGER, -- x 分量
    YCoordinate    INTEGER, -- y 分量
    HASH           OCTET STRING SIZE (32), --摘要值
    CipherText     OCTET STRING           --密文
}
```

假设使用 SM2 公钥 Q（SM2PublicKey 类型）对明文 m（字符串类型）进行加密计算。则 XCoordinate 和 YCoordinate 为随机产生的公钥的 x 分量和 y 分量。HASH 为使用 SM3 算法对明文数据运算得到的摘要值，其长度为 256 位，HASH=SM3（x || m || y），其中 x 和 y 为 Q 的 x 分量和 y 分量。CipherText 是与明文等长的密文。

假设使用 SM2 私钥 d（SM2PrivateKey 类型）对密文 c（SM2Cipher 类型）进行解密计算，则解密后将获得明文 m，其长度等于密文（c→CipherText）的长度。

4. SM2 签名数据格式

SM2 签名数据格式用 ASN.1 描述如下：

```

SM2Signature ::= SEQUENCE {
    R    INTEGER, -- 签名值的第一部分
    S    INTEGER, -- 签名值的第二部分
}

```

其中，R 和 S 的长度各为 256 位。

假设使用签名方私钥 d (SM2PrivateKey 类型) 对待签名数据 M (字符串类型) 进行签名计算。具体计算步骤如下：

① 预处理 1：使用签名方的用户身份标识 ID (字符串类型) 和签名方公钥 Q (SM2PublicKey 类型)，通过运算得到 Z 值 (字符串类型)。

$$Z = SM3(ENTL \parallel ID \parallel a \parallel b \parallel x_G \parallel y_G \parallel x_A \parallel y_A)$$

其中，ENTL 为 2 字节表示的 ID 的比特长度。 ID 为用户身份标识，无特殊约定情况下，长度为 16 字节，默认值从左至右依次为：0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x38, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x38。 a 和 b 为系统曲线参数。 x_G 和 y_G 为基点， x_A 和 y_A 为签名方公钥。

② 预处理 2：使用预处理 1 结果 Z 值 (字符串类型) 和待签名数据 M (字符串类型)，通过 SM3 运算得到摘要值 H (字符串类型)。

$$H = SM3(Z \parallel M)$$

③ 签名：使用预处理 2 结果 H (字符串类型) 和签名方私钥 d (SM2PrivateKey 类型)，通过签名计算得到签名结果 $sign$ (SM2Signature 类型)。

签名验证时，使用预处理 2 结果 H (字符串类型)、签名结果 $sign$ (SM2Signature 类型) 和签名方公钥 Q (SM2PublicKey 类型)，通过验签计算确定签名结果是否通过验证。

5. SM2 密钥对保护数据格式

在 SM2 密钥对传递时，需要对 SM2 密钥对进行加密保护。

SM2 密钥对的保护数据格式用 ASN.1 描述如下：

```

SM2EnvelopedKey ::= SEQUENCE {
    symAlgID          AlgorithmIdentifier, --对称密钥算法标识
    symEncryptedKey   SM2Cipher           --对称密钥密文
    Sm2PublicKey       SM2PublicKey,       --SM2 公钥
    Sm2EncryptedPrivateKey BIT STRING     --SM2 私钥密文
}

```

其中，具体的保护方法为：

- ① 产生一个对称密钥。
- ② 按对称密钥算法标识指定的算法对 SM2 私钥进行加密，得到私钥密文。若对称算法为分组算法，则其运算模式为 ECB。
- ③ 使用外部 SM2 公钥加密对称密钥得到对称密钥密文。
- ④ 将私钥密文、对称密钥密文封装到密钥对保护数据中。

第 9 章 数字证书格式

9.1 基本格式

IETF RFC 3280 规定了 X.509 数字证书的基本格式。

9.1.1 证书域组成 (Certificate)

X.509 数字证书由 3 个域组成，具体见表 9-1。

表 9-1 X.509 证书域组成

分 类	标 识	说 明
证书内容(待签名)	tbsCertificate	包含持有者公钥、持有者信息、签发者信息等
签名算法	signatureAlgorithm	包括摘要算法和公钥算法，如 sha1WithRSAEncryption，由算法标识和算法参数组成
签名值	signatureValue	使用签名算法，对证书内容 tbsCertificate 进行签名后的结果

X.509 证书域格式用 ASN.1 描述如下：

```
Certificate ::= SEQUENCE {
    tbsCertificate      TBSCertificate,
    signatureAlgorithm  AlgorithmIdentifier,
    signatureValue      BIT STRING }
AlgorithmIdentifier ::= SEQUENCE {
    Algorithm            OBJECT IDENTIFIER,
    Parameters           ANY DEFINED BY algorithm OPTIONAL }
```

9.1.2 证书内容 (tbsCertificate)

X.509 数字证书内容见表 9-2。

表 9-2 X.509 证书内容

分 类	标 识	说 明
版本号	version	用于区分证书格式版本，最新版本为 v3，缺省值为 v1
序列号	serialNumber	证书唯一标识，由签发者统一分配
签名算法	signature	必须与证书域中的签名算法相同
证书签发者	issuer	用于区分证书签发者，包含证书签发者身份信息
证书有效期	validity	由生效日期和失效日期组成
证书持有者	subject	用于区分证书持有者，包含证书持有者身份信息

(续表)

分 类	标 识	说 明
证书持有者公钥	subjectPublicKeyInfo	包含证书持有者公钥信息
证书签发者 ID	issuerUniqueID	表示证书签发者唯一标识
证书持有者 ID	subjectUniqueID	表示证书持有者唯一标识
扩展项	extensions	包含其他可扩展信息

X.509 证书内容格式用 ASN.1 描述如下：

```

TBSCertificate ::= SEQUENCE {
    version          [0] EXPLICIT Version DEFAULT v1,
    serialNumber      CertificateSerialNumber,
    signature         AlgorithmIdentifier,
    issuer            Name,
    validity          Validity,
    subject           Name,
    subjectPublicKeyInfo SubjectPublicKeyInfo,
    issuerUniqueID    [1] IMPLICIT UniqueIdentifier OPTIONAL,
                        -- If present, version MUST be v2 or v3
    subjectUniqueID   [2] IMPLICIT UniqueIdentifier OPTIONAL,
                        -- If present, version MUST be v2 or v3
    extensions        [3] EXPLICIT Extensions OPTIONAL
                        -- If present, version MUST be v3
}

```

1. 版本号 version

version 用于区分证书格式版本，最新版本为 v3，缺省值为 v1。当使用扩展项时，version=v3。当使用 issuerUniqueID 或 subjectUniqueID 时，version=v2 或 v3。如果只使用基本内容，则 version=v1 或 v2 或 v3。

version 格式用 ASN.1 描述如下：

```

version ::= INTEGER { v1(0), v2(1), v3(2) }

```

2. 序列号 serialNumber

serialNumber 是证书的唯一标识，由证书签发者统一分配。serialNumber 必须是正整数，同一个证书签发者（CA）所签发的每个证书的序列号必须不同，通过签发者和序列号可以区分每个证书。

serialNumber 格式用 ASN.1 描述如下：

```

CertificateSerialNumber ::= INTEGER

```

3. 签名算法 signature

signature 必须与证书域中的签名算法相同，即：signature=Certificate→signatureAlgorithm。
signature 格式用 ASN.1 描述如下：

```

AlgorithmIdentifier ::= SEQUENCE {

```

Algorithm OBJECT IDENTIFIER,
Parameters ANY DEFINED BY algorithm OPTIONAL }

4. 证书签发者 issuer

issuer 用于区分证书签发者，必须包含一个非空的 X.500 DN 项。DN 是 Distinguished Name 的缩写，表示可识别的名称，且 DN 项被定义为 X.501 规范中的 Name 类型。

issuer 格式用 ASN.1 描述如下：

```
Name ::= CHOICE {
    RDNSequence
}
RDNSequence ::= SEQUENCE OF RelativeDistinguishedName
RelativeDistinguishedName ::= SET OF AttributeTypeAndValue
AttributeTypeAndValue ::= SEQUENCE {
    type      AttributeType,
    value     AttributeValue }
AttributeType ::= OBJECT IDENTIFIER
AttributeValue ::= ANY DEFINED BY AttributeType
```

Name 类型采用分层结构，由多个属性 AttributeTypeAndValue 组成；每个属性由属性类型 AttributeType 和属性值 AttributeValue 组成；AttributeValue 的具体类型由 AttributeType 决定，通常采用 DirectoryString 类型。DirectoryString 用 ASN.1 描述如下：

```
DirectoryString ::= CHOICE {
    teletexString      TeletexString (SIZE (1..MAX)),
    printableString    PrintableString (SIZE (1..MAX)),
    universalString     UniversalString (SIZE (1..MAX)),
    utf8String         UTF8String (SIZE (1..MAX)),
    bmpString          BMPString (SIZE (1..MAX)) }
```

issuer 中的属性值应优先采用 UTF8String 编码。X.500 系列规范中定义了属性的标准集合，X.509 数字证书只使用其中的一部分属性。issuer 和 subject 包含的主要属性类型如表 9-3 所示。

表 9-3 issuer 和 subject 包含的主要属性类型

分 类	OID	说 明
country	id-at 6	国家，C
organization	id-at 10	单位，O
organization unit	id-at 11	部门，OU
distinguished name qualifier	id-at 46	DN 限定符
state or province name	id-at 8	省份或州，ST
common name	id-at 3	通用名称，CN
serial number	id-at 5	序列号，SN
locality	id-at 7	城市，L
domain component	0.9.2342.19200300.100.1.25	域名组件，等同于 DNS，DC
title	id-at 12	头衔

(续表)

分 类	OID	说 明
surname	id-at 4	姓
given name	id-at 42	名
initials	id-at 43	首字母缩写
pseudonym	id-at 65	假名
generation qualifier	id-at 44	时代限定符, 如老、小、第四代等
email address	pkcs-9 1	电子邮箱

注: id-at OBJECT IDENTIFIER ::= { joint-iso-ccitt(2) ds(5) 4 }

pkcs-9 OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840) rsadsi(113549)
pkcs(1) 9 }

当需要比较两个 issuer 或 subject 是否相同时, 应对其包含的所有属性进行比较。针对属性类型相同的属性值, 按照以下规则进行比较:

① 采用不同方式编码的属性值代表不同的字符串。如, "Marianne Swanson" 分别采用 PrintableString 和 BMPString 编码后, 代表不同的字符串。

② 除 PrintableString 编码方式外, 其他编码方式的属性值均大小写相关。可按照二进制对象进行比较。

③ PrintableString 编码方式的属性值大小写无关。如 "Marianne Swanson" 与 "MARIANNE SWANSON" 表示相同的字符串。

④ 对于 PrintableString 编码方式的属性值, 比较前应删除字符串首尾的所有空格和中间冗余的空格 (即将多个连续的空格转换为单个空格)。

5. 证书持有者 subject

subject 用于区分证书持有者。证书持有者的名称或姓名可以包含在 subject 中, 也可以包含在 subjectAltName 扩展项中。

subject 格式用 ASN.1 描述与 issuer 相同。

如果证书持有者为 CA 或 CRL 签发者, subject 必须包含一个非空的 DN 项。如果证书持有者名称或姓名只包含在 subjectAltName 扩展项中, subject 必须为一个空的 SEQUENCE, 同时 subjectAltName 扩展项必须设置为关键项 (critical=TRUE)。当 subject 为非空时, 只能包含 X.500 DN 项。同一个 CA 可以为相同的证书持有者签发多个证书, 这些证书具有相同的 subject。

subject 包含的主要属性类型与 issuer 相同。此外, subject 还可以包含 EmailAddress 属性。由于 PrintableString 字符集不包括字母 "@", 因此 EmailAddress 属性值的编码方式采用 IA5String, 包括字母 "@", 且 EmailAddress 属性值大小写无关 (如 fanfeedback@redsox.com 与 FANFEEDBACK@REDSOX.COM 表示相同的电子邮箱地址)。

6. 证书有效期 validity

validity 用于表示证书有效期, 由生效日期和失效日期组成。

validity 格式用 ASN.1 描述如下:

Validity ::= SEQUENCE {

```

notBefore      Time,
notAfter       Time }
Time ::= CHOICE {
    utcTime      UTCTime,
    generalTime  GeneralizedTime }

```

证书有效期可采用两种格式：GeneralizedTime 和 UTCTime。2050 年及之后的日期必须采用 GeneralizedTime 格式，之前的日期可采用 UTCTime 格式。

UTCTime 表示 Universal Time，是一种标准的 ASN.1 时间类型。UTCTime 用 2 位数字表示年份，时间可精确到分或秒。可包含 Z，用于表示 Greenwich Mean Time，也可包含时差。当 validity 中的 notBefore 和 notAfter 采用 UTCTime 格式时，必须采用 Greenwich Mean Time，且必须精确到秒，如 YYMMDDhhmmssZ。当 YY 小于 50 时，年份应该解释为 20YY 年，当 YY 大于或等于 50 时，年份应该解释为 19YY 年。

GeneralizedTime 表示 Generalized Time，也是一种标准的 ASN.1 时间类型。GeneralizedTime 用 4 位数字表示年份。可包含 Z 表示 Greenwich Mean Time，也可包含 Greenwich Mean Time 与本地时间的时差。当 validity 中的 notBefore 和 notAfter 采用 GeneralizedTime 格式时，必须采用 Greenwich Mean Time，且必须精确到秒，如 YYYYMMDDhhmmssZ。

7. 证书持有者公钥 subjectPublicKeyInfo

subjectPublicKeyInfo 表示证书持有者公钥信息。

subjectPublicKeyInfo 格式用 ASN.1 描述如下：

```

SubjectPublicKeyInfo ::= SEQUENCE {
    algorithm      AlgorithmIdentifier,
    subjectPublicKey BIT STRING }

```

8. 证书签发者 ID issuerUniqueID 与证书持有者 ID subjectUniqueID

issuerUniqueID 表示证书签发者的唯一标识，subjectUniqueID 表示证书持有者的唯一标识。

issuerUniqueID 和 subjectUniqueID 格式用 ASN.1 描述如下：

```

UniqueIdentifier ::= BIT STRING

```

issuerUniqueID 和 subjectUniqueID 主要用于兼容 v2 版本证书格式，只能出现在版本 v2 或 v3 格式中，在版本 v1 格式中不允许出现。由于 X.509 数字证书不允许不同的证书持有者使用相同的 DN 项，因此版本 v3 格式中不建议使用 issuerUniqueID 和 subjectUniqueID。

9. 扩展项 extensions

extensions 用于证书信息扩展，可包含多个扩展信息。

extensions 格式用 ASN.1 描述如下：

```

Extension ::= SEQUENCE SIZE (1..MAX) OF Extension
Extension ::= SEQUENCE {
    extnID      OBJECT IDENTIFIER,
    critical    BOOLEAN DEFAULT FALSE,
    extnValue    OCTET STRING }

```

extensions 只能出现在版本 v3 格式中。

每个扩展项都可设置为关键项（critical=TRUE）或非关键项（critical=FALSE）。如果遇到未知的关键扩展项，则必须拒绝该证书；如果遇到未知的非关键扩展项，可以忽略该扩展项。

每个扩展项由一个 OID 和一个 ASN.1 结构组成。OID 赋值给 extnID，ASN.1 编码后的结构赋值给 extnValue。单个证书最多只能包含特定扩展项的单个实例，如最多只能包含一个 AuthorityKeyIdentifier 扩展项。

9.2 标准扩展项

IETF RFC 3280 规定了 X.509 数字证书的标准扩展项和专用互联网扩展项。

9.2.1 标准扩展项（Standard Extensions）

X.509 数字证书的标准扩展项见表 9-4。

表 9-4 X.509 数字证书标准扩展项

/	扩 展 项	OID	critical	说 明
1	AuthorityKeyIdentifier	id-ce 35	FALSE	证书签发者密钥标识
2	SubjectKeyIdentifier	id-ce 14	TRUE	证书持有者密钥标识
3	KeyUsage	id-ce 15	TRUE	密钥用途
4	PrivateKeyUsagePeriod	id-ce 16	FALSE	私钥有效期
5	CertificatePolicies	id-ce 32		证书策略
6	PolicyMappings	id-ce 33	FALSE	策略映射
7	SubjectAltName	id-ce 17		证书持有者别名
8	IssuerAltName	id-ce 18	FALSE	证书签发者别名
9	SubjectDirectoryAttributes	id-ce 9	FALSE	证书持有者目录属性
10	BasicConstraints	id-ce 19		基本限制
11	NameConstraints	id-ce 30	TRUE	名称限制
12	PolicyConstraints	id-ce 36		策略限制
13	ExtendedKeyUsage	id-ce 37		扩展密钥用途
14	CRLDistributionPoints	id-ce 31	FALSE	CRL 发布点
15	InhibitAnyPolicy	id-ce 54	TRUE	禁止任意策略
16	FreshestCRL(DeltaCRL DistributionPoint)	id-ce 46	FALSE	最新 CRL 或增量 CRL
17	NetscapeCertType	2.16.840.1.113730.1.1	FALSE	Netscape 证书类型

注：id-ce OBJECT IDENTIFIER ::= { joint-iso-ccitt(2) ds(5) 29 }
id-pkix OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) }

1. authorityKeyIdentifier

authorityKeyIdentifier 扩展项用于区分证书签发者（CA）的公钥。当证书签发者拥有

多个公私钥对用于签发用户证书时，必须使用该扩展项。

该扩展项必须设置为非关键项（critical=FALSE）。

authorityKeyIdentifier 格式用 ASN.1 描述如下：

```
id-ce-authorityKeyIdentifier OBJECT IDENTIFIER ::= { id-ce 35 }
authorityKeyIdentifier ::= SEQUENCE {
    keyIdentifier          [0] KeyIdentifier          OPTIONAL,
    authorityCertIssuer    [1] GeneralNames          OPTIONAL,
    authorityCertSerialNumber [2] CertificateSerialNumber OPTIONAL }
KeyIdentifier ::= OCTET STRING
```

authorityKeyIdentifier 基于证书签发者证书（CA 证书）中的内容生成，主要有两种生成方式：基于 subjectKeyIdentifier，以及基于 issuer 和 serialNumber。当基于 subjectKeyIdentifier 生成时，keyIdentifier 通常等于证书签发者证书中的 subjectKeyIdentifier。

2. subjectKeyIdentifier

subjectKeyIdentifier 扩展项用于区分证书持有者的公钥。

该扩展项必须设置为关键项（critical=TRUE）。

subjectKeyIdentifier 格式用 ASN.1 描述如下：

```
id-ce-subjectKeyIdentifier OBJECT IDENTIFIER ::= { id-ce 14 }
SubjectKeyIdentifier ::= KeyIdentifier
```

subjectKeyIdentifier 可以基于公钥产生，两种常用的产生方法如下：

① 将 subjectPublicKey 删除标识(tag)、长度(length)和无用比特个数(number of unused bits)后，使用 SHA1 算法计算获得 160 比特摘要值。subjectKeyIdentifier=160 比特摘要值。

② 同方法①计算获得 160 比特摘要值。subjectKeyIdentifier=4 比特类型（'0100'）+ 至少 80 比特摘要值。

subjectKeyIdentifier 也可以使用递增的整数来表示。

3. keyUsage

keyUsage 扩展项用于定义证书中的公钥及其对应私钥的用途。当需要限制或约束密钥只能用于部分操作时，可以使用该扩展项。例如，当只允许 RSA 密钥用于验证数据签名，不允许用于验证数字证书或 CRL 中的签名时，需要将 keyUsage 设置为 digitalSignature 或 nonRepudiation；当只允许 RSA 密钥用于密钥管理时，需要将 keyUsage 设置为 keyEncipherment。

该扩展项必须设置为关键项（critical=TRUE）。

keyUsage 格式用 ASN.1 描述如下：

```
id-ce-keyUsage OBJECT IDENTIFIER ::= { id-ce 15 }
keyUsage ::= BIT STRING {
    digitalSignature      (0),
    nonRepudiation       (1),
    keyEncipherment      (2),
```

dataEncipherment	(3),
keyAgreement	(4),
keyCertSign	(5),
cRLSign	(6),
encipherOnly	(7),
decipherOnly	(8) }

其中, `digitalSignature` 表示数字签名服务, 可用于实体身份认证和数据完整性认证, 但不可用于签发证书和 CRL。

`nonRepudiation` 表示抗抵赖服务, 可用于操作或交易的抗抵赖, 但不可用于证书和 CRL 签发行行为的抗抵赖。

`keyEncipherment` 用于密钥加密。当 RSA 密钥用于密钥管理时, 需要将 `keyUsage` 设置为 `keyEncipherment`。

`dataEncipherment` 用于数据加密, 但不可用于密钥加密。

`keyAgreement` 用于密钥协商。如使用 DH 算法密钥进行密钥管理时, 需要将 `keyUsage` 设置为 `keyAgreement`。

`keyCertSign` 用于签发和验证数字证书。当 `KeyUsage` 设置为 `keyCertSign` 时, 需同时设置扩展项 `basicConstraints`→`cA=TRUE`。

`cRLSign` 用于签发和验证 CRL (包括 CRL、增量 CRL、ARL 等)。

`encipherOnly` 表示只用于数据加密。仅当 `keyAgreement` 设置时, `encipherOnly` 才有效, 表示密钥只用于密钥协商过程中的数据加密。

`decipherOnly` 表示只用于数据解密。仅当 `keyAgreement` 设置时, `decipherOnly` 才有效, 表示密钥只用于密钥协商过程中的数据解密。

4. `privateKeyUsagePeriod`

`privateKeyUsagePeriod` 扩展项用于定义私钥有效期, 允许私钥有效期不同于证书有效期。该扩展项主要用于限制签名密钥, 即与证书对应的私钥不允许在私钥有效期之外进行数字签名操作。

该扩展项必须设置为非关键项 (`critical=FALSE`)。

`privateKeyUsagePeriod` 格式用 ASN.1 描述如下:

```
id-ce-privateKeyUsagePeriod OBJECT IDENTIFIER ::= { id-ce 16 }
privateKeyUsagePeriod ::= SEQUENCE {
    notBefore      [0]    GeneralizedTime OPTIONAL,
    notAfter       [1]    GeneralizedTime OPTIONAL }
```

其中, `notBefore` 和 `notAfter` 采用 `GeneralizedTime` 格式。

5. `certificatePolicies`

`certificatePolicies` 扩展项可包括多个证书策略; 每个证书策略由一个 OID 和多个限定语 (`qualifier`) 组成。`qualifier` 是可选的, 但不能与策略的定义或内涵发生冲突。

`certificatePolicies` 格式用 ASN.1 描述如下:

```

id-ce-certificatePolicies OBJECT IDENTIFIER ::= { id-ce 32 }
anyPolicy OBJECT IDENTIFIER ::= { id-ce-certificate-policies 0 }
certificatePolicies ::= SEQUENCE SIZE (1..MAX) OF PolicyInformation
PolicyInformation ::= SEQUENCE {
    policyIdentifier    CertPolicyId,
    policyQualifiers    SEQUENCE SIZE (1..MAX) OF
        PolicyQualifierInfo OPTIONAL }
CertPolicyId ::= OBJECT IDENTIFIER
PolicyQualifierInfo ::= SEQUENCE {
    policyQualifierId    PolicyQualifierId,
    qualifier            ANY DEFINED BY policyQualifierId }
id-qt                OBJECT IDENTIFIER ::= { id-pkix 2 }
id-qt-cps            OBJECT IDENTIFIER ::= { id-qt 1 }
id-qt-unotice        OBJECT IDENTIFIER ::= { id-qt 2 }
PolicyQualifierId ::= OBJECT IDENTIFIER ( id-qt-cps | id-qt-unotice )
Qualifier ::= CHOICE {
    cPSuri            CPSuri,
    userNotice        UserNotice }
CPSuri ::= IA5String
UserNotice ::= SEQUENCE {
    noticeRef          NoticeReference OPTIONAL,
    explicitText        DisplayText OPTIONAL}
NoticeReference ::= SEQUENCE {
    organization        DisplayText,
    noticeNumbers        SEQUENCE OF INTEGER }
DisplayText ::= CHOICE {
    ia5String            IA5String            (SIZE (1..200)),
    visibleString        VisibleString        (SIZE (1..200)),
    bmpString            BMPString            (SIZE (1..200)),
    utf8String            UTF8String            (SIZE (1..200)) }

```

其中，策略限定语类型（qualifier type）主要包括 2 类：CPS Pointer 和 User Notice。

CPS Pointer 表示 CPS 指针，包含一个 URL，可链接到 CA 中心发布的 CPS (Certification Practice Statement)。

User Notice 表示用户通知内容，应用系统应显示给用户阅读，可由 2 个可选字段组成：noticeRef 和 explicitText。noticeRef 包含组织名称和通知编号，应用系统可据此获得通知内容（例如，以文件形式保存所有通知，通过编号查找文件并获得通知内容），并显示给用户阅读。explicitText 包含文本内容，最大长度不应超过 200 字符。当 noticeRef 和 explicitText 同时存在时，应优先使用 noticeRef；如无法通过 noticeRef 获得通知内容，则使用 explicitText。

6. policyMappings

policyMappings 扩展项只用于 CA 证书，可包含多对 OID，每对 OID 由 issuerDomainPolicy

和 subjectDomainPolicy 组成, 表示 issuer 域的证书策略 issuerDomainPolicy 等同于 subject 域的证书策略 subjectDomainPolicy。该扩展项中不允许包含证书策略 anyPolicy。

该扩展项必须设置为非关键项 (critical=FALSE)。

policyMappings 格式用 ASN.1 描述如下:

```
id-ce-policyMappings OBJECT IDENTIFIER ::= { id-ce 33 }
PolicyMappings ::= SEQUENCE SIZE (1..MAX) OF SEQUENCE {
    issuerDomainPolicy      CertPolicyId,
    subjectDomainPolicy     CertPolicyId }
```

7. subjectAltName

subjectAltName 扩展项表示证书持有者的别名, 可包含多个。别名形式包括电子邮箱、DNS 名称、IP 地址、URI 等, 其中 DNS 名称也可以使用 subject 中的 DN 项 domainComponent 表示。

当 subject 为空时, 该扩展项必须设置为关键项 (critical=TRUE)。

subjectAltName 格式用 ASN.1 描述如下:

```
id-ce-subjectAltName OBJECT IDENTIFIER ::= { id-ce 17 }
subjectAltName ::= GeneralNames
GeneralNames ::= SEQUENCE SIZE (1..MAX) OF GeneralName
GeneralName ::= CHOICE {
    otherName                [0]    OtherName,
    rfc822Name                [1]    IA5String,
    dNSName                   [2]    IA5String,
    x400Address                [3]    ORAddress,
    directoryName              [4]    Name,
    ediPartyName               [5]    EDIPartyName,
    uniformResourceIdentifier  [6]    IA5String,
    iPAddress                  [7]    OCTET STRING,
    registeredID               [8]    OBJECT IDENTIFIER }
OtherName ::= SEQUENCE {
    type-id    OBJECT IDENTIFIER,
    value      [0] EXPLICIT ANY DEFINED BY type-id }
EDIPartyName ::= SEQUENCE {
    nameAssigner    [0]    DirectoryString OPTIONAL,
    partyName        [1]    DirectoryString }
```

当 subjectAltName 包含电子邮箱时, 格式必须符合 rfc822Name 类型 (RFC 822 规范)。

当 subjectAltName 包含 IP 地址时, 格式必须采用网络字节序和 LSB (least significant bit) 方式, 且 IP 地址字节与 ASN.1 编码后的 OCTET 一一对应。对于 IPv4 (RFC 791 规范), 必须包含 4 个字节; 对于 IPv6 (RFC 1883 规范), 必须包含 16 个字节。

当 subjectAltName 包含 DNS 名称时, 格式必须符合 dNSName 类型 (RFC 1034 规范)。

当 subjectAltName 包含 URI 地址时, 格式必须符合 uniformResourceIdentifier 类型 (RFC

1738 规范), 不允许使用相对地址, 只能使用绝对地址。在 URI 地址中, 只有模式名 (scheme name, 如 http) 和主机名 (hostname, 如 www.sina.com) 大小写无关, 其他组成部分均大小写相关。

当 subjectAltName 包含 DN 项时, 格式必须符合 directoryName 类型, 且同一证书持有者必须具有唯一性。

8. issuerAltName

issuerAltName 扩展项表示证书签发者的别名, 可包含多个。具体要求同 subjectAltName。该扩展项必须设置为非关键项 (critical=FALSE)。

issuerAltName 格式用 ASN.1 描述如下:

```
id-ce-issuerAltName OBJECT IDENTIFIER ::= { id-ce 18 }
issuerAltName ::= GeneralNames
```

9. subjectDirectoryAttributes

subjectDirectoryAttributes 扩展项可包含证书持有者的目录属性。

该扩展项必须设置为非关键项 (critical=FALSE)。

subjectDirectoryAttributes 格式用 ASN.1 描述如下:

```
id-ce-subjectDirectoryAttributes OBJECT IDENTIFIER ::= { id-ce 9 }
subjectDirectoryAttributes ::= SEQUENCE SIZE (1..MAX) OF Attribute
```

10. basicConstraints

basicConstraints 扩展项用于区分证书持有者是否是 CA。如果是 CA, 则限制其认证路径的最大长度。

当用于终端实体证书时, 该扩展项可以设置为关键项或非关键项。当用于签发用户证书的 CA 证书时, 该扩展项必须设置为关键项 (critical=TRUE)。当 CA 证书不用于签发用户证书, 只用于签发 CRL 时, 该扩展项可以设置为关键项或非关键项。

basicConstraints 格式用 ASN.1 描述如下:

```
id-ce-basicConstraints OBJECT IDENTIFIER ::= { id-ce 19 }
basicConstraints ::= SEQUENCE {
    cA                      BOOLEAN DEFAULT FALSE,
    pathLenConstraint       INTEGER (0..MAX) OPTIONAL }
```

其中, cA 表示该证书是否是 CA。如果 cA 设置为 FALSE, 则 keyUsage 扩展项不能包含 keyCertSign。

仅当 cA 设置为 TRUE, 且 keyUsage 扩展项包含 keyCertSign 时, pathLenConstraint 才有效, 表示该 CA 证书之后认证路径中非自签名 CA 证书的最大数目 (即认证路径或信任链中, 该 CA 证书和终端实体证书之间的非自签名证书的最大数目)。pathLenConstraint 出现时必须大于或等于 0, 当等于 0 时表示该 CA 证书不能签发下级 CA 证书, 只能签发终端实体 (End Entity) 证书; 如果 pathLenConstraint 没有出现, 表明认证路径的长度没有限制。

11. nameConstraints

nameConstraints 扩展项只用于 CA 证书, 包含一个命名空间, 用于限制认证路径中后续证书中的 subject 内容和 subjectAltName 扩展项。

该扩展项必须设置为关键项 (critical=TRUE)。

nameConstraints 格式用 ASN.1 描述如下:

```
id-ce-nameConstraints OBJECT IDENTIFIER ::= { id-ce 30 }
nameConstraints ::= SEQUENCE {
    permittedSubtrees      [0]      GeneralSubtrees OPTIONAL,
    excludedSubtrees       [1]      GeneralSubtrees OPTIONAL }
GeneralSubtrees ::= SEQUENCE SIZE (1..MAX) OF GeneralSubtree
GeneralSubtree ::= SEQUENCE {
    base                    GeneralName,
    minimum                 [0]      BaseDistance DEFAULT 0,
    maximum                 [1]      BaseDistance OPTIONAL }
BaseDistance ::= INTEGER (0..MAX)
```

其中, permittedSubtrees 表示允许的名称范围, excludedSubtrees 表示无效的名称范围。permittedSubtrees 和 excludedSubtrees 可包含一个或多个命名子树; 每个命名子树定义为 GeneralName 类型, 可以采用多种形式, 如 URI、电子邮箱、DNS、IP 地址等。

如果采用 URI 形式, 该扩展项只用于限制名称的主机部分, 可以设定一个主机或域名。如果该扩展项以 “.” 开始, 则表示该域名可以扩展子域; 如 abc.xyz.com 和 abc.def.xyz.com 都属于 “.xyz.com” 扩展, 但 xyz.com 不属于 “.xyz.com” 扩展。如果该扩展项不以 “.” 开始, 则表示特定主机, 不允许任何扩展, 如 foo.bar.com 就不能扩展。

如果采用电子邮箱形式, 该扩展项可以设定一个特定邮箱、一个特定主机的所有邮箱或一个域名范围的所有邮箱。若需设定一个特定邮箱, 则直接使用完整的电子邮箱地址, 如 root@xyz.com。若需设定一个特定主机的所有邮箱, 则应使用主机名, 如 xyz.com 表示主机 xyz.com 上的所有邮箱地址。如需设定一个域名范围的所有邮箱, 应使用以 “.” 开始的域名, 如 .xyz.com 表示 xyz.com 域范围内的所有邮箱地址, 但不包括主机 xyz.com 上的邮箱地址。

如果采用 DNS 形式, 该扩展项可直接使用域名, 如 foo.bar.com; 该域名扩展后的任何新域名均满足要求。如 www.foo.bar.com 是 foo.bar.com 的扩展, 但 foo1.bar.com 就不是。

如果采用 IP 地址形式, 该扩展项必须使用 CIDR 格式 (即 “地址+掩码”, RFC 1519 规范), 用于表示 IP 地址范围。对于 IPv4 地址, 应编码成 8 个 OCTET; 如 C 类地址 10.9.8.0, CIDR 表示为 10.9.8.0/255.255.255.0, 编码后为 0A 09 08 00 FF FF FF 00。对于 IPv6, 应编码成 32 个 OCTET。

12. policyConstraints

policyConstraints 扩展项只用于 CA 证书, 用于禁止策略映射, 或用于要求认证路径中所有证书必须包含一个认可的策略 ID (policy identifier)。

该扩展项可以设置为关键项或非关键项。

policyConstraints 格式用 ASN.1 描述如下：

```
id-ce-policyConstraints OBJECT IDENTIFIER ::= { id-ce 36 }
policyConstraints ::= SEQUENCE {
    requireExplicitPolicy          [0] SkipCerts OPTIONAL,
    inhibitPolicyMapping          [1] SkipCerts OPTIONAL }
SkipCerts ::= INTEGER (0..MAX)
```

其中，如果 inhibitPolicyMapping 存在，其值 n 表示认证路径中该证书后面允许策略映射的证书数目，也就是说，认证路径中从该证书后的第 $n+1$ 个证书开始不再允许策略映射。例如，inhibitPolicyMapping 为 1 表示认证路径中，该证书签发的下级证书允许策略映射，但其他后续证书不允许策略映射。

如果 requireExplicitPolicy 存在，其值 m 表示认证路径中该证书后面不需要显性策略（explicit policy）的证书数目，也就是说，认证路径中从该证书后的第 $m+1$ 个证书开始需要显性策略。当需要一个显性策略时，应在 certificatePolicies 扩展项中包含一个认可的策略 ID。

13. extendedKeyUsage

extendedKeyUsage 扩展项用于表示证书中公钥及其对应私钥的一个或多个用途，是 keyUsage 扩展项中基本用途的替代或补充。通常，该扩展项只用于终端实体（end entity）证书。

该扩展项可以设置为关键项或非关键项，由证书签发者决定。当设置为 anyExtendedKeyUsage 时，该扩展项应该设置为非关键项（critical=FALSE）。

extendedKeyUsage 格式用 ASN.1 描述如下：

```
id-ce-extKeyUsage OBJECT IDENTIFIER ::= { id-ce 37 }
ExtKeyUsageSyntax ::= SEQUENCE SIZE (1..MAX) OF KeyPurposeId
KeyPurposeId ::= OBJECT IDENTIFIER
```

当 keyUsage 和 extendedKeyUsage 同时存在时，必须分别进行处理，该证书只能用于 keyUsage 和 extendedKeyUsage 同时允许的用途。如果 keyUsage 和 extendedKeyUsage 所定义的用途完全互斥，没有同时允许的用途，则该证书被认为无效，不能用于任何目的。

常用的扩展密钥用途用 ASN.1 描述如下：

```
anyExtendedKeyUsage OBJECT IDENTIFIER ::= { id-ce-extKeyUsage 0 }
id-kp OBJECT IDENTIFIER ::= { id-pkix 3 }
id-kp-serverAuth      OBJECT IDENTIFIER ::= { id-kp 1 }
id-kp-clientAuth      OBJECT IDENTIFIER ::= { id-kp 2 }
id-kp-codeSigning     OBJECT IDENTIFIER ::= { id-kp 3 }
id-kp-emailProtection OBJECT IDENTIFIER ::= { id-kp 4 }
id-kp-timeStamping    OBJECT IDENTIFIER ::= { id-kp 8 }
id-kp-OCSPSigning     OBJECT IDENTIFIER ::= { id-kp 9 }
```

其中, anyExtendedKeyUsage 表示所有用途。id-kp-serverAuth 表示 Web 服务器 SSL/TLS 身份认证, 等同于 keyUsage 中的 digitalSignature、keyEncipherment、keyAgreement。id-kp-clientAuth 表示 Web 客户端 SSL/TLS 身份认证, 等同于 keyUsage 中的 digitalSignature、keyAgreement。id-kp-codeSigning 表示可执行程序的代码签名, 等同于 keyUsage 中的 digitalSignature。id-kp-emailProtection 表示电子邮件保护, 等同于 keyUsage 中的 digitalSignature、nonRepudiation、keyEncipherment 或 keyAgreement。id-kp-timeStamping 表示时间戳, 即将某对象摘要值与时间绑定, 等同于 keyUsage 中的 digitalSignature、nonRepudiation。id-kp-OCSPSigning 表示 OCSP 响应包签名, 等同于 keyUsage 中的 digitalSignature、nonRepudiation。

14. cRLDistributionPoints

cRLDistributionPoints 扩展项用于确定如何获得 CRL 信息。

该扩展项应该设置为非关键项 (critical=FALSE)。

cRLDistributionPoints 格式用 ASN.1 描述如下:

```
id-ce-cRLDistributionPoints OBJECT IDENTIFIER ::= { id-ce 31 }
cRLDistributionPoints ::= SEQUENCE SIZE (1..MAX) OF DistributionPoint
DistributionPoint ::= SEQUENCE {
    distributionPoint      [0]      DistributionPointName OPTIONAL,
    reasons                [1]      ReasonFlags OPTIONAL,
    cRLIssuer              [2]      GeneralNames OPTIONAL }
DistributionPointName ::= CHOICE {
    fullName              [0]      GeneralNames,
    nameRelativeToCRLIssuer [1]    RelativeDistinguishedName }
ReasonFlags ::= BIT STRING {
    unused                (0),
    keyCompromise         (1),      --表示密钥泄露
    cACompromise          (2),      --表示 CA 泄露
    affiliationChanged    (3),      --表示关系变更
    superseded            (4),      --表示废弃
    cessationOfOperation  (5),      --表示操作中止
    certificateHold        (6),      --表示证书冻结
    privilegeWithdrawn     (7),      --表示权限撤销
    aACompromise          (8) }     --表示 AA 泄露
```

其中, DistributionPoint 类型不能只包含 reasons 字段, distributionPoint 和 cRLIssuer 字段至少包含一个。如果证书签发者不是 CRL 签发者, 则 cRLIssuer 字段必须存在, 且必须包含 CRL 签发者的 DN 名称。如果证书签发者也是 CRL 签发者, 则 cRLIssuer 字段必须忽略, distributionPoint 字段必须存在。

distributionPoint 定义为 DistributionPointName 类型。如果 distributionPoint 包含多个名称值 (GeneralNames), 则每个名称表示一种方法或机制, 不同方法或机制能获取相同的 CRL, 如 LDAP 和 HTTP。如果 distributionPoint 只包含单个值 (nameRelativeToCRLIssuer),

则表示 DN 项的一部分；只需将其附加到 CRL 签发者或证书签发者的 X.500 DN 名称后即可获得 CRL 发布点名称；如果 DistributionPoint 中的 cRLIssuer 存在，则使用 CRL 签发者 cRLIssuer，否则使用证书签发者。

15. inhibitAnyPolicy

inhibitAnyPolicy 扩展项只用于 CA 证书，用于表示认证路径中哪些证书允许 anyPolicy 策略（OID 为 2.5.29.32.0）。

该扩展项应该设置为关键项（critical=TRUE）。

inhibitAnyPolicy 格式用 ASN.1 描述如下：

```
id-ce-inhibitAnyPolicy OBJECT IDENTIFIER ::= { id-ce 54 }
inhibitAnyPolicy ::= SkipCerts
SkipCerts ::= INTEGER (0..MAX)
```

其中，SkipCerts 表示认证路径中该证书后面允许 anyPolicy 策略的证书数目，也就是说，认证路径中从该证书后的第 SkipCerts+1 个证书开始不再允许 anyPolicy 策略。例如，SkipCerts 为 1 表示认证路径中该证书签发的下级证书允许 anyPolicy 策略，但其他后续证书不允许 anyPolicy 策略。

16. freshestCRL (Delta CRL Distribution Point)

freshestCRL 扩展项用于确定如何获取增量 CRL 信息。该扩展项格式与 cRLDistributionPoints 扩展项完全一致。

该扩展项应该设置为非关键项（critical=FALSE）。

freshestCRL 格式用 ASN.1 描述如下：

```
id-ce-freshestCRL OBJECT IDENTIFIER ::= { id-ce 46 }
FreshestCRL ::= CRLDistributionPoints
```

17. netscapeCertType

netscapeCertType 扩展项表示 Netscape 证书类型，用于定义证书中公钥及其对应私钥的一个或多个用途，与 extendedKeyUsage 扩展项的功能类似。

该扩展项必须设置为非关键项（critical=FALSE）。

netscapeCertType 格式用 ASN.1 描述如下：

```
id-NetscapeCertType OBJECT IDENTIFIER ::= { 2.16.840.1.113730.1.1 }
netscapeCertType ::= BIT STRING {
    SSLClient                (0),
    SSLServer                (1),
    S/MIME                   (2),
    Object Signing           (3),
    Reserved                 (4),
    SSL CA                   (5),
    S/MIME CA                (6),
    Object Signing CA        (7) }
```

9.2.2 专用互联网扩展项

X.509 数字证书的专用互联网扩展项（Private Internet Extensions）见表 9-5。

表 9-5 X.509 数字证书专用互联网扩展项

/	扩展项	OID	critical	说明
1	AuthorityInfoAccess	id-pe 1	FALSE	证书签发者信息访问
2	SubjectInfoAccess	id-pe 11	FALSE	证书持有者信息访问

注：id-pe OBJECT IDENTIFIER ::= { id-pkix 1 }

1. authorityInfoAccess

authorityInfoAccess 扩展项用于确定如何访问证书签发者（CA）的信息和服务，如在线验证服务、CA 策略数据（不包含 CRL 访问地址，对于 CRL 访问地址应使用 cRLDistributionPoints 扩展项）等。

该扩展项可以用于 CA 证书，也可以用于终端实体（End Entity）证书，但必须设置为非关键项（critical=FALSE）。

authorityInfoAccess 格式用 ASN.1 描述如下：

```
id-pe-authorityInfoAccess OBJECT IDENTIFIER ::= { id-pe 1 }
authorityInfoAccessSyntax ::= SEQUENCE SIZE (1..MAX) OF AccessDescription
AccessDescription ::= SEQUENCE {
    accessMethod      OBJECT IDENTIFIER,
    accessLocation     GeneralName }
```

其中，accessMethod 字段表示信息格式和类型，accessLocation 字段表示信息地址。访问机制或方式可由 accessMethod 确定，也可由 accessLocation 确定。

常用的 accessMethod 用 ASN.1 描述如下：

```
id-ad OBJECT IDENTIFIER ::= { id-pkix 48 }
id-ad-caIssuers OBJECT IDENTIFIER ::= { id-ad 2 }
id-ad-ocsp OBJECT IDENTIFIER ::= { id-ad 1 }
```

当 accessMethod 为 id-ad-ocsp 时，accessLocation 表示 OCSP 服务器地址，通过 OCSP 服务可获得当前证书的作废状态。

当 accessMethod 为 id-ad-caIssuers 时，accessLocation 表示服务器地址和访问协议，可获得参考描述（referenced description）信息。accessLocation 定义为 GeneralName 类型，可以采用多种形式。当通过 http、ftp 或 ldap 方式访问信息时，accessLocation 必须为 URI。当通过 DAP（Directory Access Protocol）方式访问信息时，accessLocation 必须为目录名称 directoryName，且该目录名称入口应在 crossCertificatePair 属性中包含 CA 证书。当通过 email 方式访问信息时，accessLocation 必须为 rfc822Name。

2. subjectInfoAccess

subjectInfoAccess 扩展项用于确定如何访问证书持有者的信息和服务。如果证书持有

者 **subject** 是 CA，信息和服务包括证书验证服务、CA 策略数据等；如果证书持有者是终端实体（End Entity），则描述服务类型和访问方法。

该扩展项可以用于 CA 证书，也可以用于终端实体证书，但必须设置为非关键项（critical=FALSE）。

subjectInfoAccess 格式用 ASN.1 描述如下：

```
id-pe-subjectInfoAccess OBJECT IDENTIFIER ::= { id-pe 11 }
subjectInfoAccessSyntax ::= SEQUENCE SIZE (1..MAX) OF AccessDescription
AccessDescription ::= SEQUENCE {
    accessMethod          OBJECT IDENTIFIER,
    accessLocation        GeneralName }

```

其中，**accessMethod** 字段表示信息格式和类型，**accessLocation** 字段表示信息地址。访问机制或方式可由 **accessMethod** 确定，也可由 **accessLocation** 确定。

常用的 **accessMethod** 用 ASN.1 描述如下：

```
id-ad OBJECT IDENTIFIER ::= { id-pkix 48 }
id-ad-caRepository OBJECT IDENTIFIER ::= { id-ad 5 }
id-ad-timeStamping OBJECT IDENTIFIER ::= { id-ad 3 }

```

当证书持有者 **subject** 是 CA 时，**accessMethod** 可以使用 **id-ad-caRepository**，表示该 CA 通过 **repository** 发布其签发的证书和 CRL。**accessLocation** 定义为 **GeneralName** 类型，可以采用多种形式。当通过 **http**、**ftp** 或 **ldap** 方式访问信息时，**accessLocation** 必须为 **URI**；当通过 **DAP**（Directory Access Protocol）方式访问信息时，**accessLocation** 必须为目录名称 **directoryName**；当通过 **email** 方式访问信息时，**accessLocation** 必须为 **rfc822Name**。

当证书持有者采用 **TSP** 协议提供时间戳服务时，**accessMethod** 可以使用 **id-ad-timeStamping**。当通过 **http**、**ftp** 或 **ldap** 方式访问时间戳服务时，**accessLocation** 必须为 **URI**。当通过 **email** 方式访问时间戳服务时，**accessLocation** 必须为 **rfc822Name**。当通过 **TCP/IP** 方式访问时间戳服务时，**accessLocation** 可以采用域名或 IP 地址。

9.3 国内扩展项

9.3.1 卫生系统专用扩展项

《卫生系统数字证书格式规范》规定了卫生系统数字证书的专用扩展项，见表 9-6。

表 9-6 卫生系统数字证书专用扩展项

/	扩展项	OID	critical	说明
1	SubjectUniqueID	自行定义	FALSE	证书持有者唯一标识

1. subjectUniqueID

subjectUniqueID 扩展项代表一个证书持有者身份的唯一编码（实体唯一标识），在业务系统中，本标识可与应用系统内的用户账号一一关联，从而用于实现证书用户与系统用

户的绑定。该扩展项 OID 由各电子认证服务机构自行申请和定义。

对于终端实体证书，该扩展项必须签发，该项应为非关键扩展项。

subjectUniqueID 的编码规范如下：

用户编号（变长）+ “@” + CA 编号（4 位）+ 证件类型代码（2 位）+ 安全标识（1 位）+ 证件号码（变长）

其中，用户编号是一个证书持有者的证书序号，建议用户编号采用阿拉伯数字。例如一个企业申请 2 个证书，则第一张证书的用户编号为 1，第 2 张证书的用户编号为 2，依次类推。

CA 编号应为 CA 机构的《电子认证服务许可证》上“许可证编号”的后四位数字。

证件类型代码是证书用户申请数字证书使用的关键证件的编码，证书类型和号码类型的代码如表 9-7 所示。

表 9-7 证书类型与证件类型代码对应表

证 书 类 型	办理证书时可使用的证件名称	证件类型代码
内部机构证书	组织机构代码/工商营业执照/税务登记证/其他	JJ/GS/SW/QT
内部工作人员证书	身份证/军官证/护照/回乡证/其他	SF/JG/HZ/HX/QT
内部设备证书	组织机构代码/工商营业执照/税务登记证/其他	JJ/GS/SW/QT
外部机构证书	组织机构代码/工商营业执照/税务登记证/其他	JJ/GS/SW/QT
个人证书	身份证/军官证/护照/回乡证/其他	SF/JG/HZ/HX/QT
外部设备证书	组织机构代码/工商营业执照/税务登记证/其他	JJ/GS/SW/QT

安全标识使用 1 位数字代表不同含义，其意义如下：

0：代表其后的“证件号码”为明文格式签发；

1：代表其后的“证件号码”为 Base64 编码格式签发；

用户根据不同的证书类型，应提供不同的证件办理数字证书。

例如，个人证书办理时提供的证件为身份证，身份证号码为：342222197205053618，CA 机构编号为 1001，该 CA 中心为用户签发的第一张数字证书的实体唯一标识应为：

① 安全标识为 0 时的值应为：1@1001SF0342222197205053618

② 安全标识为 1 时的值应为：1@1001SF1MzQyMjlyMTk3MjA1MDUzNjE4，其中 SF1 后面的内容为 Base64(342222197205053618)的结果。

对于机构证书，如果提供组织机构代码证件号码为：123456789，CA 机构编号为 1001，该 CA 中心为用户签发的第一张数字证书的实体唯一标识应为：

① 安全标识为 0 时的值应为：1@1001JJ0123456789

② 安全标识为 1 时的值应为：1@1001JJ1MTIzNDU2Nzg5，其中 JJ1 后面的内容为 Base64(123456789)结果。

实体唯一标识数据总长度不应超过 128 字节，属性编码应使用 UTF8String。

9.3.2 国内通用扩展项

《基于 SM2 密码算法的数字证书格式规范》规定了数字证书的国内通用扩展项，见表 9-8。

表 9-8 数字证书国内通用扩展项

/	扩展项	OID	是否关键项	说明
1	IdentifyCode	1.2.156.10260.4.1.1	FALSE	个人身份标识码
2	InsuranceNumber	1.2.156.10260.4.1.2	FALSE	个人社会保险号
3	ICRegistrationNumber	1.2.156.10260.4.1.3	FALSE	企业工商注册号
4	OrganizationCode	1.2.156.10260.4.1.4	FALSE	企业组织机构代码
5	TaxationNumber	1.2.156.10260.4.1.5	FALSE	企业税号

1. identifyCode

identifyCode 扩展项用于表示个人的身份标识号码。该扩展项应该设置为非关键项 (critical=FALSE)。

identifyCode 格式用 ASN.1 描述如下：

```
id-IdentifyCode OBJECT IDENTIFIER ::= { 1.2.156.10260.4.1.1 }
identifyCode ::= CHOICE {
    residentifierCardNumber [0] PrintableString OPTIONAL,
    militaryOfficerCardNumber [1] UTF8String OPTIONAL,
    passportNumber [2] PrintableString OPTIONAL }
```

其中，residentifierCardNumber 表示身份证号码，militaryOfficerCardNumber 表示军官证号码，passportNumber 表示护照号码。

2. insuranceNumber

insuranceNumber 扩展项用于表示个人的社会保险号码。该扩展项应该设置为非关键项 (critical=FALSE)。

insuranceNumber 格式用 ASN.1 描述如下：

```
id-InsuranceNumber OBJECT IDENTIFIER ::= { 1.2.156.10260.4.1.2 }
insuranceNumber ::= PrintableString
```

3. iCRegistrationNumber

iCRegistrationNumber 扩展项用于表示企业的工商注册号。该扩展项应该设置为非关键项 (critical=FALSE)。

iCRegistrationNumber 格式用 ASN.1 描述如下：

```
id-ICRegistrationNumber OBJECT IDENTIFIER ::= { 1.2.156.10260.4.1.3 }
iCRegistrationNumber ::= PrintableString
```

4. organizationCode

organizationCode 扩展项用于表示企业的组织机构代码。该扩展项应该设置为非关键项 (critical=FALSE)。

organizationCode 格式用 ASN.1 描述如下：

```
id-OrganizationCode OBJECT IDENTIFIER ::= { 1.2.156.10260.4.1.4 }
organizationCode ::= PrintableString
```

5. taxationNumber

taxationNumber 扩展项用于表示企业税号码。该扩展项应该设置为非关键项 (critical=FALSE)。

taxationNumber 格式用 ASN.1 描述如下：

```
id-TaxationNumber OBJECT IDENTIFIER ::= { 1.2.156.10260.4.1.5 }
```

```
taxationNumber ::= PrintableString
```

第 10 章 数字证书分类

为适应复杂的应用场景，在实际应用中，通常需要将证书进行分类。有些分类方式 X.509 格式已经支持，但有些分类方式 X.509 格式本身并不支持，需要通过其他方式来识别。

证书通常可分为两大类：根据证书持有者分类和根据密钥分类。

10.1 根据证书持有者分类

1. 根据证书持有者是否为 CA 进行分类

根据证书持有者是否为 CA，可将证书分为 2 类：CA 证书和用户证书。CA 证书可以给用户或其他 CA 签发证书，用户证书不允许给其他用户或 CA 签发证书。

X.509 格式中通过扩展项 BasicConstraints 来区分这 2 类证书。当其中的 cA 项为 TRUE 时表示 CA 证书，为 FALSE 时表示用户证书。

BasicConstraints 扩展项格式用 ASN.1 描述如下：

```
BasicConstraints ::= SEQUENCE {  
    cA                               BOOLEAN DEFAULT FALSE,  
    pathLenConstraint               INTEGER (0..MAX) OPTIONAL }
```

2. 按照证书持有者类型进行分类

根据证书持有者类型，通常将证书分为几类：个人证书、单位证书和系统证书等。

个人证书是 CA 系统给个人签发的证书，代表个人身份。证书中需要包含个人信息（如姓名、身份证、E-mail、电话等）和个人的公钥。

单位证书是 CA 系统给机构或组织等签发的证书，代表单位身份。证书中需要包含单位信息（如名称、组织机构代码、E-mail、联系人等）和单位的公钥。

系统证书是 CA 系统给软件系统或设备系统等签发的证书，代表系统身份。证书中需要包含系统信息（如 IP 地址、域名等）和系统的公钥。系统证书又包括 Web 服务器证书、域控制器证书、VPN 设备证书、OCSP 服务器证书、时间戳服务器证书等。

X.509 格式本身并不支持这种分类，通常通过在 Subject 中增加 DN 项进行区分，如可增加 OU=PERSON 表示个人证书，OU=UNIT 表示单位证书等。为保持证书内容的统一性，扩展项 KeyUsage、ExtKeyUsage 必须设置合适的值。

10.2 根据密钥分类

1. 根据密钥对产生方式进行分类

根据密钥对的产生方式，可将证书分为 2 类：签名证书和加密证书。

签名证书及私钥只用于签名验签，不能用于加密解密。为保证该密钥对的唯一性，该密钥对必须由用户端密码模块产生和保存，在证书签发过程中 CA 中心并不知道其私钥，只对其公钥进行操作。

加密证书及私钥只用于加密解密，不能用于签名验签。为实现密钥恢复或行业监管，该密钥对必须由 CA 中心产生，并回送给用户端密码模块保存。CA 中心同时保存该密钥对，必要时可恢复该密钥对。

X.509 格式本身并不支持这种分类；通常通过存储位置或应用系统进行区分。为保持证书内容的统一性，扩展项 KeyUsage、ExtKeyUsage 必须设置合适的值。

KeyUsage 中已经定义的类型如下。

- ① digitalSignature: 表示数字签名；
- ② nonRepudiation: 表示不可抵赖；
- ③ keyEncipherment: 表示密钥加密；
- ④ dataEncipherment: 表示数据加密；
- ⑤ keyAgreement: 表示密钥协商；
- ⑥ keyCertSign: 表示证书签名；
- ⑦ cRLSign: 表示 CRL 签名；
- ⑧ encipherOnly: 表示只用于加密；
- ⑨ decipherOnly: 表示只用于解密。

2. 根据证书用途进行分类

根据证书用途，通常将证书分为 SSL 服务器证书、SSL 客户端证书、代码签名证书、Email 证书、时间戳服务器证书、OCSP 服务器证书等。SSL 证书只用于 SSL/TLS 应用，Email 证书只用于安全电子邮件，代码签名证书只用于对代码进行签名验签。

X.509 格式中通过扩展项 ExtKeyUsage 来区分这几类证书。为保持证书内容的统一性，扩展项 KeyUsage 必须设置合适的值。

ExtKeyUsage 中已经定义的类型如下。

- ① id-kp-serverAuth: 用于 SSL/TLS Web 服务器身份认证；
- ② id-kp-clientAuth: 用于 SSL/TLS Web 客户端身份认证；
- ③ id-kp-codeSigning: 用于对可下载的执行代码进行签名；
- ④ id-kp-emailProtection: 用于保护 E-mail；
- ⑤ id-kp-timeStamping: 用于将对象摘要值与时间绑定；
- ⑥ id-kp-OCSPSigning: 用于对 OCSP 响应包进行签名。

第 11 章 私钥与证书存储方式

公钥无需保密，包含在数字证书中，并以数字证书形式对外公开发布，由数字证书的安全机制来保证公钥的完整性和真实性。

私钥必须保密，有多种存储方式，且不同的存储方式安全性不同。

通常将密码算法运算、密钥存储及产生的装置称为密码模块。密码模块又可分为软件密码模块和硬件密码模块。

11.1 证书保存形式

11.1.1 DER 文件形式

数字证书按照 Certificate 类型进行 DER 编码后，以二进制形式保存为文件。

当证书采用 DER 文件形式保存时，常用的文件后缀为 der 或 crt。

例如，用户 ZHANG San 的数字证书内容显示如图 11-1 所示，其中序列号=1174 (0x0496)，证书签发者 DN=“CN = Virtual CA, C = CN”，证书持有者 DN=“CN = ZHANG San, OU = Person, C = CN”，证书有效期=20130222000000-20160222000000。

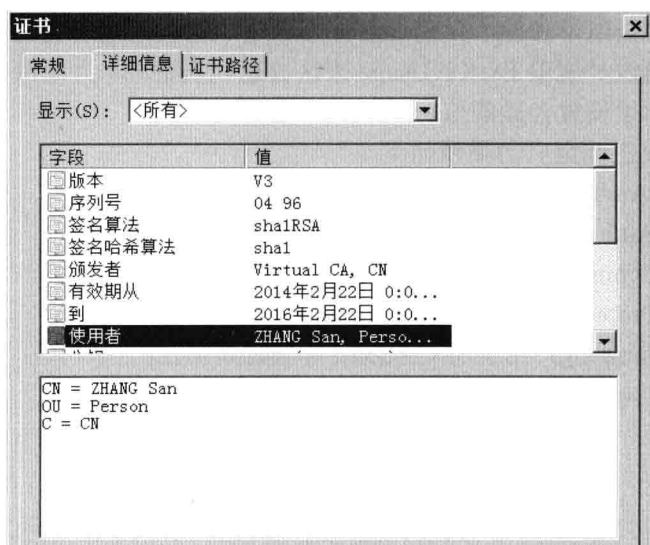


图 11-1 数字证书内容

该数字证书 DER 编码后的文件大小为 752 字节，具体二进制值如表 11-1 所示，其中，每行显示 16 个字节，每行最前面 4 个数字表示该行第 1 个字节的地址序号（从 0 开始）。

表 11-1 数字证书 DER 文件内容

```

0000: 30 82 02 EC 30 82 01 D4 -- A0 03 02 01 02 02 02 04
0010: 96 30 0D 06 09 2A 86 48 -- 86 F7 0D 01 01 05 05 00
0020: 30 22 31 0B 30 09 06 03 -- 55 04 06 13 02 43 4E 31
0030: 13 30 11 06 03 55 04 03 -- 13 0A 56 69 72 74 75 61
0040: 6C 20 43 41 30 1E 17 0D -- 31 34 30 32 32 31 31 36
0050: 30 30 30 30 5A 17 0D 31 -- 36 30 32 32 31 31 36 30
0060: 30 30 30 5A 30 32 31 0B -- 30 09 06 03 55 04 06 13
0070: 02 43 4E 31 0F 30 0D 06 -- 03 55 04 0B 13 06 50 65
0080: 72 73 6F 6E 31 12 30 10 -- 06 03 55 04 03 13 09 5A
0090: 48 41 4E 47 20 53 61 6E -- 30 81 9F 30 0D 06 09 2A
00A0: 86 48 86 F7 0D 01 01 01 -- 05 00 03 81 8D 00 30 81
00B0: 89 02 81 81 00 B4 F6 CF -- 18 3D 5E 8E 1D 46 7A 90
00C0: 7D 8E 41 D2 E3 C8 F1 A3 -- AE F3 6D 8A 24 FF 55 23
00D0: 25 BD EB 0C D0 7B 87 36 -- 5D 1F 73 98 65 3E 57 97
00E0: F6 65 7D 13 E0 E1 B5 FC -- BC 38 6F 56 3E 57 4E D6
00F0: 51 1D 13 12 7C 33 B3 60 -- 31 79 32 07 97 F3 3C 8B
0100: 29 0D B5 78 38 93 CE 84 -- E4 A3 DD FB F9 25 47 1C
0110: 72 A6 5E 78 02 CF F3 48 -- 9D CA D9 00 73 DE 4B 16
0120: 07 52 48 20 06 F3 4F CA -- A5 2D 66 88 95 C6 6C D6
0130: 3F 61 34 F7 E3 02 03 01 -- 00 01 A3 81 9F 30 81 9C
0140: 30 0C 06 03 55 1D 13 01 -- 01 FF 04 02 30 00 30 1D
0150: 06 03 55 1D 0E 04 16 04 -- 14 2C 04 87 10 60 FC 61
0160: F6 2B 64 81 3D FB 66 30 -- DA F0 73 BC 08 30 0E 06
0170: 03 55 1D 0F 01 01 FF 04 -- 04 03 02 03 F8 30 29 06
0180: 03 55 1D 25 04 22 30 20 -- 06 08 2B 06 01 05 05 07
0190: 03 02 06 0A 2B 06 01 04 -- 01 82 37 14 02 02 06 08
01A0: 2B 06 01 05 05 07 03 04 -- 30 11 06 09 60 86 48 01
01B0: 86 F8 42 01 01 04 04 03 -- 02 05 A0 30 1F 06 03 55
01C0: 1D 23 04 18 30 16 80 14 -- 96 F0 94 F8 49 8D 23 05
01D0: 86 B0 CA B5 2D 7A 9A 60 -- 32 FB B0 F9 30 0D 06 09
01E0: 2A 86 48 86 F7 0D 01 01 -- 05 05 00 03 82 01 01 00
01F0: 8D 42 AD 5C DF C7 C7 90 -- FA 58 C0 74 15 C6 4F 20
0200: 9B F1 49 9C B8 3C 22 98 -- 45 75 A6 0D 7C 02 9D 83
0210: 1D C4 5D CF 4F 8E 57 E7 -- 0A 9B 67 02 33 23 59 76
0220: B4 B5 B7 F3 27 36 6F F4 -- 32 6C 1C E9 B3 4B 81 DC
0230: D0 CF 2E CF 07 4C 65 75 -- 74 DF 23 9D 7D 2B E4 F1
0240: 15 0C 84 61 41 5F DC 67 -- 92 A9 7C 39 A0 CA A9 58
0250: 6B ED 7D 94 08 F7 83 42 -- 61 F8 62 D8 DC 3B 5D B7
0260: 69 5C D0 36 F2 99 A8 0C -- 99 6E B0 0C 21 E3 98 9F
0270: 12 6D D1 76 4E 0C 31 CB -- 7F 54 73 FE 96 83 76 35
0280: 22 2F BF F6 2B 11 04 3A -- A7 BE 33 3C D5 DA EE 56
0290: 7A C4 1A 67 3B 77 DE 52 -- C0 DA 09 CA 45 71 11 B2
02A0: D5 35 BF 44 54 08 C2 FA -- 0C 5C EF C0 EF 82 63 37
02B0: 3C 4C AB 59 4C FD 6C 2A -- 9D 64 27 35 4E 4F D8 2E
02C0: 2C 5C EB A1 99 DB FA 3A -- 53 54 13 92 91 5D 8F 38
02D0: DD 1C D8 AB 34 22 9A EF -- 8A E4 62 C2 23 9D 06 A5
02E0: D7 D8 58 B7 F4 98 CA 61 -- 29 9D DE A8 F6 DA CC 81

```

将该表内容恢复成二进制文件，取文件后缀为 cer 或 crt，在 Windows 环境下鼠标双击即可查看该数字证书的内容。

11.1.2 Base64 文件形式

由于数字证书 DER 编码后的内容为二进制形式，不方便显示，因此需要将其转换成文本形式，通常采用 Base64 编码方式。

例如，表 11-1 中所示 ZHANG San 的数字证书 Base64 编码后，长度由 752 字节变成 1004 字符，如表 11-2 所示。

表 11-2 数字证书 Base64 文件内容

```
MIIC7DCCAdSgAwIBAgICBJYwDQYJKoZIhvcNAQEFBQAwIjELMAkGA1UEBhMCQ04x
EzARBgNVBAMTCiZpcnR1YVWwgQ0EwHhcNMTQwMjIxMTYwMDAwWheNMTYwMjIxMTYw
MDAwWjAyMQswCQYDVQQUJDEtJDEPMA0GA1UECXMUGUGVyc29uMRIwEAYDVQQDEwla
SEFORyBTYw4wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBALT2zxg9Xo4dRnqQ
fY5B0uPI8aOu822KJP9VIyW96wzQe4c2XR9zmGU+V5f2ZX0T4OG1/Lw4b1Y+V07W
UR0TEnwzs2AxeTIHl/M8iykNtXg4k86E5Kpd+/klRxxyp154As/zSJ3K2QBz3ksW
B1JIIAbzT8qLLWallcZs1j9hNPfjAgMBAAGjgZ8wgZwwDAYDVDR0TAQH/BAIwADAd
BgNVHQ4EFgQULASHEGD8YfYrZIE9+2Yw2vBzvAgwDgYDVDR0PAQH/BAQDAGP4MCKG
A1UdJQQIMCAGCCsGAQUFBwMCBgorBgEEAYI3FAICBggrBgEFBQcDBDARBglghkgB
hvhCAQEEBAMCBaAwHwYDVDR0jBBgwFoAUlvCU+EmNIwWGsMq1LXqaYDL7sPkWdQYYJ
KoZIhvcNAQEFBQADggEBAl1CrVzf8eQ+lJAdBXGTyCb8UmcuDwimEV1pg18Ap2D
HcRdz0+OV+cKm2cCMYnZdrS1t/MnNm/0Mmwc6bNLgdzQzy7PB0xldXTf1519K+Tx
FQyEYUf3GeSqXw5oMqpWGvtfZQI94NCYfhi2Nw7XbdpXNA28pmoDJJusAwh45if
Em3Rdk4MMct/VHP+loN2NSIvv/YrEQQ6p74zPNXa7lZ6xBpnO3feUsDaCcpFeRGy
1TW/RFQIwvoMXO/A74JjNzxMq1lM/WwqnWQnNU5P2C4sXOuhmdv6OINUE5KRXY84
3RzYqzQimu+K5GLCI50GpdfYWLf0mMphKZ3eqPbazIE=
```

将该表内容恢复成文本文件，取文件后缀为 cer 或 crt，在 Windows 环境下鼠标双击即可查看该数字证书的内容。

11.1.3 PKCS#7 文件形式

为方便交换证书链（证书路径或认证路径）上的所有证书，需要将多个证书保存到单个文件中，通常采用 PKCS#7 编码形式。

当证书链采用 PKCS#7 文件形式保存时，常用的文件后缀为 p7b。

PKCS#7 定义了多种密码消息形式，主要包括：data、signedData、envelopedData、signedAndEnvelopedData、digestData、encryptedData、keyAgreementInfo 等。当用于保存证书链时，具体要求如下：

1. ContentInfo

```
ContentInfo ::= SEQUENCE {
    contentType ContentType,
```


content [0] EXPLICIT ANY DEFINED BY contentType OPTIONAL }

其中，contentType = signedData，content 为 SignedData 类型。

2. SignedData

SignedData ::= SEQUENCE {
 version Version,
 digestAlgorithms DigestAlgorithmIdentifiers,
 contentInfo ContentInfo,
 certificates [0] IMPLICIT ExtendedCertificatesAndCertificates OPTIONAL,
 crls [1] IMPLICIT CertificateRevocationLists OPTIONAL,
 signerInfos SignerInfos }

其中，digestAlgorithms 为空；contentInfo→contentType=data，contentInfo→content 可为空；certificates 包含证书链中的证书；crls 可忽略，signerInfos 可为空。

11.1.4 Windows 证书库形式

为加强对数字证书的分类管理，方便应用系统使用，方便用户操作和查看，Windows 提供了证书库机制。

1. 证书库分类

Windows 证书库分为以下几类，如图 11-2 所示。



图 11-2 Windows 证书库

① 受信任的根证书颁发机构。保存多个可信任的根 CA 证书。通过该类证书可以验证用户证书或子 CA 证书的合法性。

② 中级证书颁发机构。保存多个子 CA 证书。

③ 受信任的发布者。保存多个可信任的可执行代码发布者证书。如果某可执行程序具有代码签名，且使用该类证书能验证代码签名的合法性，则说明该程序值得信赖。

④ 未受信任的发布者。保存多个不受信任的可执行代码发布者证书。如果某可执行程序具有代码签名，且使用该类证书能验证代码签名的合法性，则说明该程序不受信任，可能存在安全风险，不建议安装或使用。

⑤ 其他人证书。保存多个他人的证书。

⑥ 个人证书。包含自己的数字证书。如果有对应的私钥，并与数字证书进行关联。

2. 证书库管理

打开 IE 浏览器，单击菜单“Internet 选项”后进入“内容”页，如图 11-3 所示。

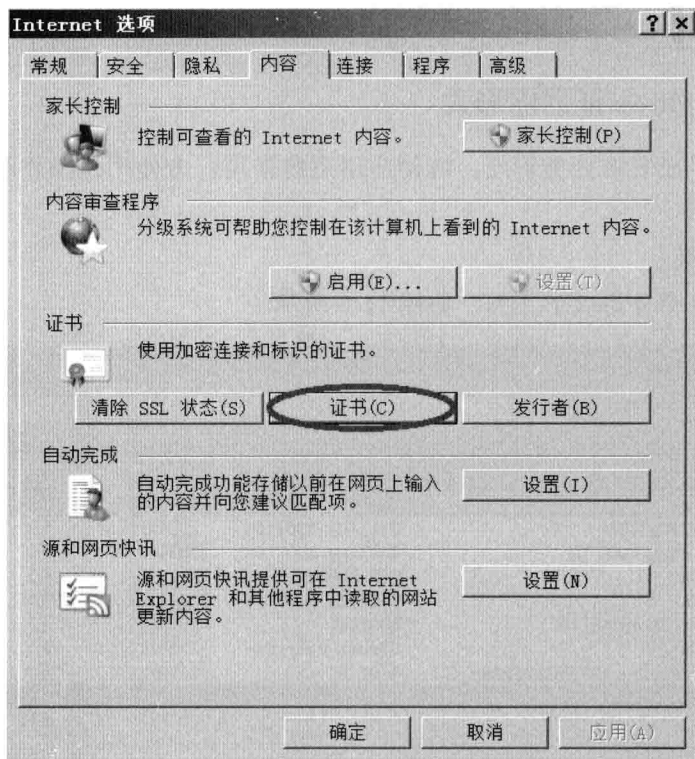


图 11-3 IE 浏览器 Internet 选项

单击按钮“证书”后即可进入证书库管理界面，如图 11-2 所示。通过手工可以导入、导出、删除各类证书。

可将证书库中的证书导出为文件形式，导出过程中可选择导出的文件格式：DER 编码二进制格式、Base64 编码格式、PKCS#7 格式，如图 11-4 所示。

其中，导出个人证书时，可以选择是否将私钥跟证书一起导出。如果选择导出私钥，则只能导出为 PKCS#12 格式文件。

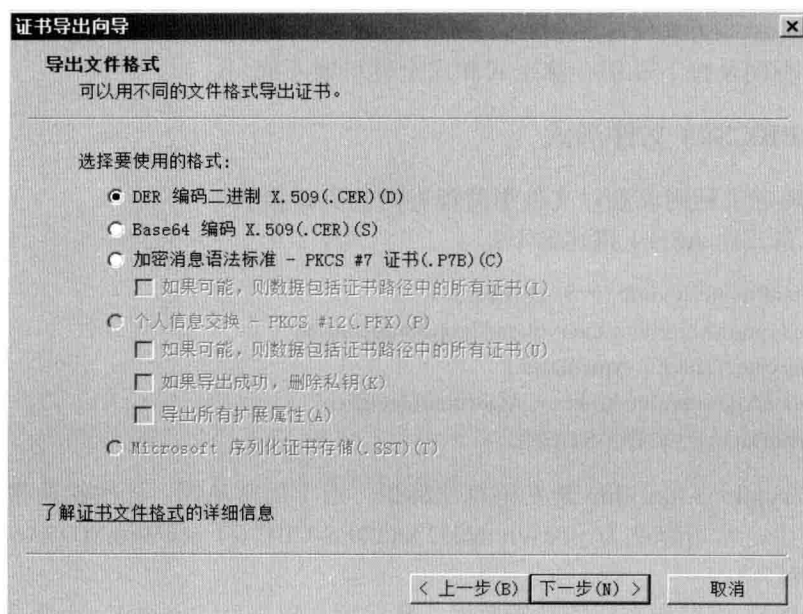


图 11-4 Windows 证书库导出证书格式

3. 证书库访问

Windows CryptoAPI 包含证书库管理函数, 允许应用系统直接访问证书库, 主要包括以下 API 函数。

- ① CertOpenStore: 根据证书库类型打开证书库;
- ② CertCloseStore: 关闭证书库;
- ③ CertEnumCertificatesInStore: 从证书库中枚举证书;
- ④ CertFindCertificateInStore: 从证书库中查找指定的证书;
- ⑤ CertCreateCertificateContext: 根据证书数据创建证书句柄;
- ⑥ CertFreeCertificateContext: 释放证书句柄;
- ⑦ CertGetCertificateContextProperty: 获取证书句柄属性;
- ⑧ CertSetCertificateContextProperty: 设置证书句柄属性;
- ⑨ CertGetNameString: 获得证书中签发者或持有者的 DN 项。

11.2 私钥保存形式

私钥保存形式主要包括: 文件形式、密码设备形式和软件系统形式。

当采用文件形式保存私钥时, 私钥的安全性通常采用口令进行保护, 同时基于口令可对私钥文件进行加密存储。当系统使用私钥进行签名或解密时, 需要将该私钥文件读入内存或密码模块中进行密码运算。为保护私钥的安全性, 通常使用口令对私钥文件进行加密保护。

当采用密码设备形式保存私钥时, 密码设备可提供安全机制保护私钥存储的安全性和私钥访问的安全性。

当采用软件系统方式保存私钥时，私钥完全由软件系统进行管理，其安全性完全依赖于软件系统，不同系统下私钥存储形式和安全性可能不同。

11.2.1 PKCS#8 文件形式

PKCS#8 规定了私钥单独以文件形式保存时的具体格式。

私钥密文格式用 ASN.1 描述如下：

```
EncryptedPrivateKeyInfo ::= SEQUENCE {
    encryptionAlgorithm EncryptionAlgorithmIdentifier,
    encryptedData EncryptedData }
EncryptionAlgorithmIdentifier ::= AlgorithmIdentifier
EncryptedData ::= OCTET STRING
```

其中，`encryptionAlgorithm` 是密码算法标识，用于加密私钥。该密钥通常为对称密钥，可通过口令产生，常用算法为 `pbeWithMD2AndDES-CBC` 和 `pbeWithMD5AndDES-CBC`，可参见 PKCS#5。

`encryptedData` 保存加密后的私钥数据。私钥明文首先以 `PrivateKeyInfo` 形式编码，由 `encryptionAlgorithm` 所标识的对称密钥加密后保存到 `encryptedData` 中。`PrivateKeyInfo` 用 ASN.1 描述如下：

```
PrivateKeyInfo ::= SEQUENCE {
    version Version,
    privateKeyAlgorithm PrivateKeyAlgorithmIdentifier,
    privateKey PrivateKey,
    attributes [0] IMPLICIT Attributes OPTIONAL }
Version ::= INTEGER
PrivateKeyAlgorithmIdentifier ::= AlgorithmIdentifier
PrivateKey ::= OCTET STRING
Attributes ::= SET OF Attribute
```

其中，`version` 表示格式版本，缺省值为 0。`privateKeyAlgorithm` 表示私钥算法标识，如 `rsaEncryption`。`privateKey` 包含私钥明文。对于 RSA 算法，为 `RSAPrivateKey` 类型。`attributes` 为同私钥一起被加密的属性集合。

11.2.2 PKCS#12 文件形式

当私钥采用 PKCS#12 形式保存时，常用的文件后缀为 `pfx` 或 `p12`。

1. PFX

PKCS #12 规定了私钥和证书以单个文件形式保存时的具体格式。用 ASN.1 描述如下：

```
PFX ::= SEQUENCE {
    Version      INTEGER {v3(3)}(v3,...),
    authSafe     ContentInfo,
    macData      MacData OPTIONAL }
```

其中, version 表示格式版本, 缺省值为 3。

2. macData

macData 包含消息认证码 mac 及相关参数 macSalt、iterations, 采用口令方式保证 authSafe 数据完整性。用 ASN.1 描述如下:

```
MacData ::= SEQUENCE {
    mac          DigestInfo,
    macSalt      OCTET STRING,
    iterations    INTEGER DEFAULT 1 }
```

其中, mac 表示消息认证码, 用于验证 authSafe 数据的完整性。用于计算 mac 的 MAC 密钥基于口令和 macSalt、iterations 产生。mac 值使用 HMAC 算法从 authSafe 和 MAC 密钥产生。iterations 缺省值为 1, 只为兼容以前的格式, 建议使用大整数, 如 1024。

3. authSafe

authSafe 包含私钥和证书数据, 采用密码消息 ContentInfo 类型。authSafe 的 contentType 字段可以是 signedData 类型, 也可以是 data 类型。当为 signedData 类型时, 基于公钥保证数据完整性; 当采用 data 类型时, 基于口令保证完整性, 消息认证码信息保存在 macData 中。authSafe 的 content 字段直接(data 类型)或间接(signedData 类型)包含 AuthenticatedSafe 类型的编码结果。AuthenticatedSafe 用 ASN.1 描述如下:

```
AuthenticatedSafe ::= SEQUENCE OF ContentInfo
```

其中, ContentInfo 的 content 字段可以是明文 Data、加密数据 EncryptedData 或数字信封数据 EnvelopedData 类型。Data 不加密, EncryptedData 基于口令加密, EnvelopedData 基于公钥加密。当使用 EncryptedData 或 EnvelopedData 类型时, 其数据的明文定义为 SafeContent 组成。

4. SafeContent

SafeContent 用 ASN.1 描述如下:

```
SafeContents ::= SEQUENCE OF SafeBag
SafeBag ::= SEQUENCE {
    bagId          BAG-TYPE.&id ({PKCS12BagSet})
    bagValue       [0] EXPLICIT BAG-TYPE.&Type ({PKCS12BagSet} {@bagId}),
    bagAttributes  SET OF PKCS12Attribute OPTIONAL }
PKCS12Attribute ::= SEQUENCE {
    attrId         ATTRIBUTE.&id ({PKCS12AttrSet}),
    attrValues     SET OF ATTRIBUTE.&Type ({PKCS12AttrSet} {@attrId}) }
PKCS12AttrSet ATTRIBUTE ::= {
    friendlyName |    -- PKCS #9 定义
    localKeyId,       --PKCS #9 定义 }
```

其中, 每个 SafeBag 只包含一个密钥或证书, 通过 OID 来区分。bagAttributes 字段允许给密钥设定昵称 (friendlyName) 或标识符 (localKeyId), 在需要时可显示给用户查看。

5. 常用 SafeBag 类型

常用 SafeBag 类型包括：keyBag、pkcs8ShroudedKeyBag、certBag、crlBag、secretBag、safeContentsBag 等，用 ASN.1 描述如下：

```

KeyBag ::= PrivateKeyInfo
PKCS8ShroudedKeyBag ::= EncryptedPrivateKeyInfo
CertBag ::= SEQUENCE {
    certId      BAG-TYPE.&id    ({CertTypes}),
    certValue   [0] EXPLICIT BAG-TYPE.&Type ({CertTypes} {@certId}) }
CRLBag ::= SEQUENCE {
    crlId      BAG-TYPE.&id    ({CRLTypes}),
    crlValue   [0] EXPLICIT BAG-TYPE.&Type ({CRLTypes} {@crlId}) }
SecretBag ::= SEQUENCE {
    secretTypeId BAG-TYPE.&id ({SecretTypes}),
    secretValue  [0] EXPLICIT BAG-TYPE.&Type ({SecretTypes} {secretTypeId}) }

```

其中，KeyBag 和 PKCS8ShroudedKeyBag 用于保存私钥。CertBag 用于保存数字证书，通过 OID 来区分证书类型，常用证书类型包括 x509Certificate 和 sdsiCertificate。CRLBag 用于保存 CRL，通过 OID 来区分 CRL 类型，目前只有一种类型 x509CRL。SecretBag 用于保存用户的个人秘密数据。

KeyBag 类型 OID 用 ASN.1 描述如下：

```

bagtypes OBJECT IDENTIFIER ::= {pkcs-12 10 1}
BAG-TYPE ::= TYPE-IDENTIFIER
keyBag      BAG-TYPE ::= {KeyBag IDENTIFIED BY {bagtypes 1}}
pkcs8ShroudedKeyBag BAG-TYPE ::= {PKCS8ShroudedKeyBag IDENTIFIED BY {bagtypes 2}}
certBag BAG-TYPE ::= {CertBag IDENTIFIED BY {bagtypes 3}}
crlBag BAG-TYPE ::= {CRLBag IDENTIFIED BY {bagtypes 4}}
secretBag BAG-TYPE ::= {SecretBag IDENTIFIED BY {bagtypes 5}}
safeContentsBag BAG-TYPE ::= {SafeContents IDENTIFIED BY {bagtypes 6}}
x509Certificate BAG-TYPE ::= {OCTET STRING IDENTIFIED BY {certTypes 1}}
sdsiCertificate BAG-TYPE ::= {IA5String IDENTIFIED BY {certTypes 2}}
x509CRL BAG-TYPE ::= {OCTET STRING IDENTIFIED BY {certTypes 1}}

```

11.2.3 Java Keystore 文件形式

JDK 1.4 已经支持 X.509 数字证书标准。Java 平台将密钥库作为密钥（公钥私钥对）和证书的资源管理库。从物理上讲，密钥库是个文件，缺省名称为.keystore。通过一个别名来区分密钥和证书，每个别名均由唯一的口令进行保护。密钥库本身也受口令保护。

当私钥采用 Java Keystore 形式保存时，常用的文件后缀为 jks。

Keystore 文件可通过两种方式进行访问或操作：keytool 工具，Java 类。

keytool 是 JDK 自带工具，可方便管理密钥库中的公钥私钥对和数字证书。主要包括以下命令：

- ① genkey：产生公私钥对。

② **import**: 导入证书或证书链。例如, `keytool -import -alias newroot -file root.cer-keystore server.jks` 表示导入别名为 `alias` 的证书文件 `root.cer`。

③ **export**: 导出证书。例如, `keytool -export -alias myssl -keystore server.jks -rfc -file server.cer` 表示将别名为 `myssl` 的证书导出为 `server.cer` 文件。

④ **certreq**: 产生 PKCS#10 格式的证书申请请求。

⑤ **storepassword**: 修改密钥库保护口令。

⑥ **keypassword**: 修改私钥保护口令。

⑦ **delete**: 删除证书。例如, `keytool -delete -keystore server.jks -alias tomcat` 表示删除别名为 `tomcat` 的证书。

⑧ **list**: 查看密钥库信息。例如, `keytool -list -v -keystore c:\server.jks` 表示查看密钥库 `server.jks` 信息。

可通过 `java.security.KeyStore` 类对密钥库中的公钥对和数字证书进行管理。主要包括以下方法。

① `Enumeration<String> aliases()`: 列出所有别名。

② `boolean containsAlias(String alias)`: 检查是否存在某别名。

③ `void deleteEntry(String alias)`: 删除别名 `alias`。

④ `Certificate getCertificate(String alias)`: 获取别名为 `alias` 的证书。

⑤ `String getCertificateAlias(Certificate cert)`: 获取证书 `cert` 的别名。

⑥ `Certificate[] getCertificateChain(String alias)`: 获取别名为 `alias` 的证书链。

⑦ `Key getKey(String alias, char[] password)`: 通过口令 `password` 获取别名为 `alias` 的密钥。

⑧ `void setCertificateEntry(String alias, Certificate cert)`: 保存证书 `cert` 到别名 `alias` 位置。

⑨ `void setKeyEntry(String alias, byte[] key, Certificate[] chain)`: 保存密钥 `key` 到别名 `alias` 位置。

⑩ `void setKeyEntry(String alias, Key key, char[] password, Certificate[] chain)`: 保存密钥 `key` 到别名 `alias` 位置, 并使用口令 `password` 保护。

11.2.4 密码设备形式

密码设备是指以硬件形式存在的密码模块。

该模式下, 私钥保存在密码设备中, 具体包括以下几个方面。

1. 密码设备分类

① 基于 IC 卡技术的密码设备。主要包括 IC 卡 (接触式和非接触式)、USB Key。IC 卡通过串口与主机连接, 采用 ISO 7816 协议进行通信; USB Key 通过 USB 与主机连接, 通过 USB 协议进行通信。

② 密码卡。通过主机主板上的扩展槽与主机连接, 采用 PCI、PCI-E 等协议进行通信。

③ 密码机。通过网络接口与主机连接, 采用 TCP/IP 协议进行通信。

2. 私钥存储方式

私钥在不同密码设备内部可能以不同形式存在。如 IC 卡或 USB Key 中, 私钥以 EF

形式存在。在加密机或加密卡中，私钥可能存储在 EPROM 芯片或专用密码芯片中。

3. 私钥存储安全性

密码设备具有很好的安全性，能有效保护私钥存储的安全性，具体措施包括：

- ① 不允许私钥以明文形式导出密码设备。
- ② 密码设备能有效防止非法强制入侵或破坏。如密码机硬件被非法强制打开时，将自动销毁所有密钥。
- ③ 不允许远程对密码设备进行配置管理，只允许通过密码机自带的管理屏幕进行操作。
- ④ 配置管理时采用高强度的身份认证手段，如口令、指纹、密钥卡。有些配置管理还要求多人共同操作。

4. 私钥访问方式

因私钥保存在密码设备中，为保证安全性，应用系统不允许直接从密码设备中取出私钥后再进行解密或签名操作，而是向密码设备发送解密或签名请求，委托密码设备间接调用私钥完成解密或签名操作。

应用系统可通过以下几种方式访问私钥：

① 报文方式。应用系统直接将待解密或签名的数据组成请求包，发送给密码设备，密码设备完成解密或签名操作后，将响应包返回给应用系统。报文协议可以采用 APDU、XML、TLV 等；APDU 适用于 IC 类密码设备，XML、TLV 等适用于密码机或密码卡。通信协议可以采用 TPDU（T=0 或 T=1）、USB、串口、PCI 或 PCIE、TCP/IP、HTTP、FTP 等。TPDU、USB、串口等适用于 IC 类密码设备，PCI 或 PCIE 等适用于密码卡，TCP/IP、HTTP、FTP 等适用于密码机。

② API 方式。应用系统直接将待解密或签名的数据提交给本地 API 接口模块，由 API 接口模块作为代理，将待解密或签名的数据组成请求包，采用报文方式发送给密码设备，密码设备完成解密或签名操作后，将响应包返回给 API 接口模块，由 API 接口模块解析响应包后，将结果数据返回给应用系统。常用的 API 规范有 PKCS#11、CrptoAPI、CAPICom、JCE、PC/SC、国密接口等。

5. 私钥访问安全性

密码设备具有很好的安全性，能有效保护私钥访问的安全性，具体措施包括：

- ① 应用系统不直接访问私钥，而是通过密码设备间接访问私钥。
- ② 应用系统访问密码设备需要进行身份认证。常用的认证方式有 IP 地址、口令、数字证书方式等。
- ③ 应用系统与密码设备之间进行通信时，可对传输数据进行加密保护。

11.2.5 软件系统形式

当采用软件系统方式保存私钥时，私钥完全由软件系统进行管理，其安全性完全依赖于软件系统，不同系统下私钥存储形式和安全性可能不同。私钥可能以单独文件形式存在，也可能与其他数据打包后保存。

Windows 系统采用 CSP（Cryptographic Service Provider）机制保存私钥。每个 CSP 都

是一个独立的密码模块，不仅可以存储私钥，而且可以执行所有的密码操作。CSP 负责创建和销毁密钥并提供各种密码操作。不同 CSP 通过名称来区别，Windows 系统已安装的 CSP 包括：Microsoft Base Cryptographic Provider v1.0、Microsoft Enhanced Cryptographic Provider v1.0、Microsoft Strong Cryptographic Provider 等，这些 CSP 均由软件实现，属于软件密码模块。

Windows CryptoAPI 包含 CSP 访问函数，允许应用系统直接访问，主要包括以下 API 函数。

- ① CryptAcquireContext: 根据 CSP 名称获得 CSP 句柄;
- ② CryptEnumProviders: 枚举系统中所有 CSP;
- ③ CryptGetDefaultProvider: 获得系统缺省 CSP;
- ④ CryptSetProvider: 设置系统缺省 CSP;
- ⑤ CryptGetProvPara: 获得 CSP 各种属性;
- ⑥ CryptSetProvParam: 设置 CSP 各种属性;
- ⑦ CryptReleaseContext: 释放 CSP 句柄。

大部分 Web 服务器已经支持数字证书机制，如 IIS、Apache、Tomcat 等，只有 Web 服务器申请并配置了 SSL 服务器证书，才能通过 TLS/SSL 服务，实现用户对网站的身份认证、用户与网站间数据的保密性、网站对用户身份认证等安全功能。但大部分 Web 服务器自身私钥的存储方式和保存格式并不公开，完全由 Web 服务器进行安全管理，且只能通过专用工具或 UI 界面进行管理和配置，不允许其他应用系统进行访问。

第 12 章 私钥与证书访问方式

12.1 CryptoAPI

12.1.1 CryptoAPI 简介

Windows CryptoAPI 是微软公司提出的安全加密应用服务框架，它提供了在 Win32 环境下使用认证、编码、加密和签名等安全服务的标准加密接口，用于增强应用程序的安全性与可控性。应用开发者在不了解复杂的加密机制和加密算法的情况下，可以简便、快速地开发出标准、通用和易于扩展的安全加密应用程序。

CryptoAPI 是一组函数，为了完成数学计算，必须具有密码服务提供者模块 (Cryptographic Service Provider, CSP)。Microsoft 通过 RSA Base Provider 在操作系统级提供一个 CSP，使用 RSA 公钥加密算法，更多的 CSP 可以根据需要增加到应用中。事实上，CSP 有可能与硬件设备（如智能卡）一起来进行数据加密。CryptoAPI 接口允许通过简单的函数调用来加密数据、交换公钥、哈希一个消息来建立摘要以及生成数字签名。CryptoAPI 还为许多高级安全性服务提供了密码操作，包括用于加密客户机/服务器消息，用于在各个平台之间传递机密数据和密钥的 PFX、代码签名等。

CryptoAPI 体系共由 5 部分组成，体系结构如图 12-1 所示。

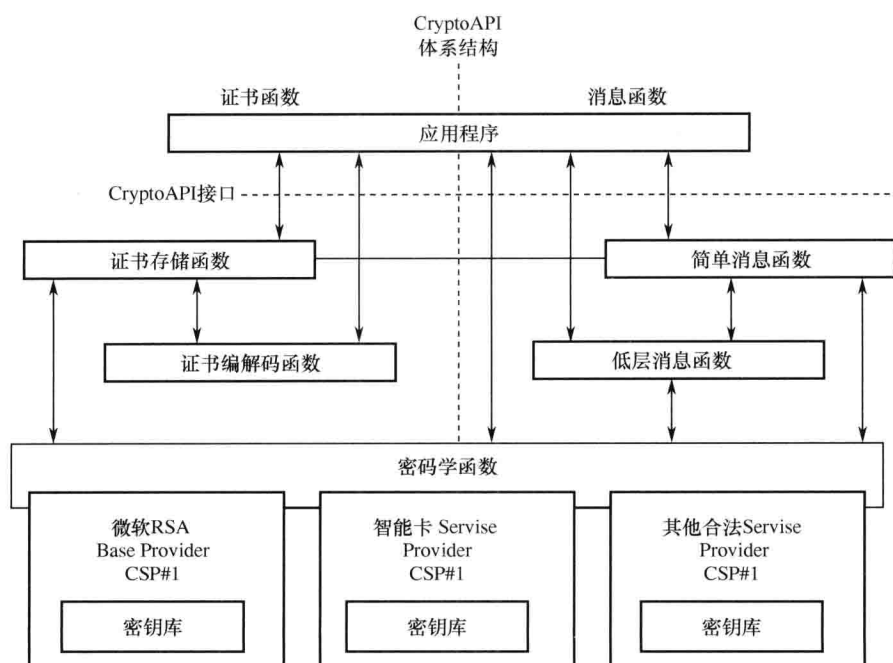


图 12-1 CryptoAPI 体系结构

(1) 基本加密函数 (Base Cryptographic Functions)

用于选择 CSP、建立 CSP 连接、产生密钥、交换及传输密钥等操作。

(2) 证书编解码函数 (Certificate Encode/Decode Functions)

用于数据加密、解密、哈希等操作。这类函数支持数据的加密/解密操作；计算哈希、签发和验证数字签名操作；实现证书、证书撤销列表、证书请求和证书扩展等编码和解码操作。

(3) 证书库管理函数 (Certificate Store Functions)

用于数字证书及证书库管理等操作。这组函数用于管理证书、证书撤销列表和证书信任列表的使用、存储、获取等。

(4) 简单消息函数 (Simplified Message Functions)

用于消息处理，比如消息编码/解码、消息加/解密、数字签名及签名验证等操作。它把多个低层消息函数包装在一起以完成某个特定任务，方便用户的使用。

(5) 低层消息函数 (Low-level Message Functions)

低层消息函数对传输的 PKCS#7 数据进行编码，对接收到的 PKCS#7 数据进行解码，并且对接收到的消息进行解码和验证。它可以实现简单消息函数的所有功能，且提供更大的灵活性，但需要更多的函数调用。

每类函数的命名前缀都有约定，前缀约定如下：基本加密函数 **Crypt**、证书编码与解码函数 **Cert**、证书库管理函数 **Store**、简单消息函数 **Message**、低层消息函数 **Msg**。

CryptoAPI 之上是应用程序，之下是 CSP。CSP 是一个真正执行加密功能的独立模块，典型的 CSP 有微软 RSA Base Provider。任何一个加密服务提供者若想成为合法的 CSP，就必须获得微软授权的一个签名文件，该签名文件保证了该 CSP 的合法性和 CryptoAPI 能够识别它。CryptoAPI 使用系统注册表存储一个 CSP 数据库，CSP 数据库中记录了所有已安装到计算机中的 CSP。

每个 CSP 都有一个名字和一个类型，CSP 的名字是唯一的，这样便于 CryptoAPI 找到对应的 CSP。目前已经有 10 种 CSP 类型，并且还会继续增长。表 12-1 列出它们支持的密钥交换算法、签名算法、对称加密算法和哈希算法。

表 12-1 CSP 类型列表

CSP 类型	交 换 算 法	签 名 算 法	对称加密算法	哈 希 算 法
PROV_RSA_FULL	RSA	RSA	RC2、RC4	MD5、SHA
PROV_RSA_AES	RSA	RSA	RC2、RC4、AES	MD5、SHA
PROV_RSA_SIG	无	RSA	无	MD5、SHA
PROV_RSA_SCHANNEL	RSA	RSA	RC4、DES、3DES	MD5、SHA
PROV_DSS	DSS	无	DSS	MD5、SHA
PROV_DSS_DH	DH	DSS	CYLINK_MEK	MD5、SHA
PROV_DH_SCHANNEL	DH	DSS	DES、3DES	MD5、SHA
PROV_FORTEZZA	KEA	DSS	Skipjack	SHA
PROV_MS_EXCHANGE	RSA	RSA	CAST	MD5
PROV_SSL	RSA	RSA	可变	可变

每个 CSP 都有一个密钥库，里面存储着由 CSP 保存的算法密钥。每个密钥库都包含

一个或多个密钥容器，每个容器都包含一到多个密钥对。每个密钥容器都被赋予一个唯一的名字，这个名字是程序要获得此容器句柄时传给函数的参数。

12.1.2 使用证书

12.1.2.1 函数说明

1. 打开证书库

```
HCERTSTORE WINAPI CertOpenSystemStore(
    HCRYPTPROV hProv,
    LPCTSTR szSubsystemProtocol);
```

参数：

hProv：CSP 句柄，如果为 NULL，则为缺省 CSP；如果不为 NULL，必须是由 CryptAcquireContext 得到的 CSP 句柄。

szSubsystemProtocol：系统证书库的名称。可以为“CA”、“MY”、“ROOT”、“SPC”。

2. 枚举证书库证书

```
PCCERT_CONTEXT WINAPI CertEnumCertificatesInStore(
    HCERTSTORE hCertStore,
    PCCERT_CONTEXT pPrevCertContext);
```

参数：

hCertStore：证书存储区的句柄。

pPrevCertContext：上一个证书内容的 CERT_CONTEXT 结构指针，当第一次开始列举时参数必须为 NULL。

说明：从证书存储区中获取第一个或者下一个证书。使用循环，它可以获取存储区所有的证书序列。

3. 获取证书名称串

```
DWORD WINAPI CertGetNameString(
    PCCERT_CONTEXT pCertContext,
    DWORD dwType,
    DWORD dwFlags,
    void* pvTypePara,
    LPTSTR pszNameString,
    DWORD cchNameString);
```

参数：

pCertContext：包含 CERT_CONTEXT 证书内容的指针。

dwType：指定如何找到名字和输出什么格式。例如 CERT_NAME_SIMPLE_DISPLAY_TYPE 会输出找到的通用名、组织单位、组织结构、E-mail。

dwFlags：指定需要处理的类型，为 CERT_NAME_ISSUER_FLAG 时，请求颁发者名字，不设置时，请求使用者名字。

pvTypePara：由参数 dwType 类型指定，是一个指向 DWORD 或 OID 对象的指针。

pszNameString: 存储转换后字符串序列。

cchNameString: pszNameString 分配的缓冲区长度, 包括结尾的 NULL 字符。

说明: 得到证书的使用者或颁发者名称, 并且把它转换成字符串。

4. 关闭证书库

```
HCERTSTORE WINAPI CertCloseStore(
    HCERTSTORE hCertStore,
    DWORD dwFlags);
```

参数:

hCertStore: 已经打开的证书库句柄。

dwFlags: 此参数的典型值为 0。缺省就是关闭证书库, 对于为上下文分配的内存并不释放。如果想要检查并且释放所有为证书、CRL 和 CTL 上下文分配的内存, 应设置下列标志: CERT_CLOSE_STORE_CHECK_FLAG (检查没有释放的证书、CRL 和 CTL 上下文)、CERT_CLOSE_STORE_FORCE_FLAG (强制释放所有和证书库相关的上下文)。

说明: 此函数释放证书库句柄。

5. 复制证书句柄

```
PCCERT_CONTEXT WINAPI CertDuplicateCertificateContext(
    PCCERT_CONTEXT pCertContext);
```

参数:

pCertContext: 包含 CERT_CONTEXT 证书内容的指针。

说明: 复制一份现有证书句柄。

6. 释放证书句柄

```
BOOL WINAPI CertFreeCertificateContext (
    PCCERT_CONTEXT pCertContext);
```

参数:

pCertContext: 包含 CERT_CONTEXT 证书内容的指针。

说明: 释放证书句柄。

12.1.2.2 示例程序

证书存储在 Windows 证书库里, 要对证书进行访问, 先使用 CertOpenSystemStore 函数打开证书库, 再使用 CertEnumCertificatesInStore 函数枚举证书, 再使用 CertGetNameString 函数获取证书名称, 使用完证书库后, 使用 CertCloseStore 函数关闭, 以防止内存泄露。

在查找到证书后, 如果需要保存留作后用, 需要使用 CertDuplicateCertificateContext 函数复制证书句柄。当不再使用此证书句柄时, 使用 CertFreeCertificateContext 释放。

1. 证书库操作

```
HCERTSTORE      hCertStore;
PCCERT_CONTEXT  pCertContext=NULL, tmpContext;
char            pszNameString[256];
```

```

DWORD          cbData;
// 第一步, 使用缺省 CSP, 打开 "MY" 密钥库
hCertStore = CertOpenSystemStore(NULL, _T("MY"));
if (hCertStore == NULL) {
    // 提示错误信息并退出
    TCHAR *errMsg = _T("不能打开证书库: MY");
    MessageBox(buf, "错误信息", MB_OK|MB_ICONERROR);
    return FALSE;
}
// 第二步, 使用 CertEnumCertificatesInStore 从打开的证书库中获得证书
// 获取库中第一个证书时, 需要把 pCertContext 置为 NULL
pCertContext = NULL;
while(pCertContext= CertEnumCertificatesInStore(hCertStore, pCertContext)) {
    // 第三步, 获取证书通用名
    cbData = CertGetNameString(pCertContext,
        CERT_NAME_SIMPLE_DISPLAY_TYPE, 0, NULL, pszNameString, 256);
    if (cbData > 1 && cbData < 256) {
        printf("\n 发现证书: %s \n", pszNameString);
    }
}
// 第四步, 关闭证书库句柄
CertCloseStore(hCertStore, 0);
return TRUE;

```

2. 复制证书句柄

```

PCCERT_CONTEXT pDupCertContext = NULL;
// 复制证书对象指针
if (pDupCertContext = CertDuplicateCertificateContext(pCertContext)) {
    printf("成功复制证书对象指针\n");
    // 使用复制的指针...
    CertFreeCertificateContext(pDupCertContext); // 释放复制的指针
}
else {
    printf("复制指针错误\n");
    exit(1);
}

```

12.1.3 使用私钥

12.1.3.1 函数说明

1. 通过证书获得私钥

```

BOOL WINAPI CryptAcquireCertificatePrivateKey(
    PCCERT_CONTEXT pCertContext,
    DWORD          dwFlags,
    void*          pvReserved,

```

```

HCRYPTPROV_OR_NCRYPT_KEY_HANDLE* phCryptProv,
DWORD* pdwKeySpec,
BOOL* pfCallerFreeProvOrN);

```

参数:

pCertContext: 包含 CERT_CONTEXT 证书内容的指针。

dwFlags:

CRYPT_ACQUIRE_CACHE_FLAG: 如果一个句柄已经缓存, 则返回相同的句柄; 否则, 获取一个新的句柄和缓存使用证书的 CERT_KEY_CONTEXT_PROP_ID 属性。若设置此标志, pfCallerFreeProvOrNCryptKey 变量接收 FALSE 时调用应用程序不能释放句柄。

CRYPT_ACQUIRE_COMPARE_KEY_FLAG: 比较证书中的公钥与 CSP 返回的公钥, 如果密钥不匹配, 获取失败, 错误代码设置为 NTE_BAD_PUBLIC_KEY。如果返回缓存句柄, 则不比较公钥。

CRYPT_ACQUIRE_NO_HEALING: 此功能将不会尝试重新创建证书上下文中的 CERT_KEY_PROV_INFO_PROP_ID 属性, 即使此属性不能被检索到。

CRYPT_ACQUIRE_SILENT_FLAG: 此环境中 CSP 显示任何用户界面。如果 CSP 必须显示 UI 以进行操作, 导致调用失败, 错误代码设置为 NTE_SILENT_CONTEXT。

CRYPT_ACQUIRE_USE_PROV_INFO_FLAG: 使用该证书的 CERT_KEY_PROV_INFO_PROP_ID 属性来判断是否应该缓存。如果以前的调用中, CRYPT_KEY_PROV_INFO 结构的 dwFlags 成员包含了 CERT_SET_KEY_CONTEXT_PROP, 此函数只使用缓存。

pvReserved: 保留, 为 NULL。

phCryptProv: 输出值, CSP 服务提供者句柄。

pdwKeySpec: 输出值, 密钥类型, 签名密钥或加密密钥。

pfCallerFreeProv: 输出值, 表明是否释放输出提供者句柄 hCryptProv。

说明: 对于指定证书上下文得到一个提供者句柄和私钥类型。

2. 创建哈希对象

```

BOOL WINAPI CryptCreateHash(
    HCRYPTPROV hProv,
    ALG_ID Algid,
    HCRYPTKEY hKey,
    DWORD dwFlags,
    HCRYPTHASH* phHash);

```

参数:

hProv: 指定容器的 CSP 模块句柄。

Algid: 哈希算法的标识符, 支持 CALG_MD5、CALG_SHA1、CALG_SHA_256、CALG_SHA_384、CALG_SHA_512。

hKey: 如果哈希算法是密钥哈希, 如 HMAC 或 MAC 算法, 就用此密钥句柄传递密钥。对于非密钥算法, 此参数为 NULL。

dwFlags: 保留。必须为 0。

phHash: 输出参数, 哈希对象的句柄。

说明：此函数初始化哈希数据流。它创建并返回一个 CSP 哈希对象的句柄，此句柄由 CryptHashData 来调用。

3. 计算数据哈希值

```
BOOL WINAPI CryptHashData(  
    HCRYPTHASH hHash,  
    BYTE* pbData,  
    DWORD dwDataLen,  
    DWORD dwFlags);
```

参数：

hHash：哈希对象句柄。

pbData：指向要计算哈希值的数据指针。

dwDataLen：数据长度。

dwFlags：对 CRYPT_USERDATA 标志，所有微软 CSP 都忽略此参数。如果所有其他 CSP 不忽略此参数，且置此参数，CSP 提示用户直接输入数据。

说明：此函数把一段数据加入到指定的哈希对象中。

4. 销毁哈希对象句柄

```
BOOL WINAPI CryptDestroyHash(  
    HCRYPTHASH hHash);
```

参数：

hHash：哈希对象句柄。

说明：此函数销毁由参数指定的哈希对象。当一个哈希对象被销毁后，它对程序来说将不可用。

5. 计算数据签名值

```
BOOL WINAPI CryptSignHash(  
    HCRYPTHASH hHash,  
    DWORD dwKeySpec,  
    LPCTSTR sDescription,  
    DWORD dwFlags,  
    BYTE* pbSignature,  
    DWORD* pdwSigLen);
```

参数：

hHash：哈希对象句柄。

dwKeySpec：CSP 容器使用的密钥类型，签名时使用 AT_SIGNATURE。

sDescription：不再使用，必须置为 NULL。

dwFlags：一般为 0，若与 RSA 密钥提供者一起使用，可设置为 CRYPT_NOHASHOID 时，表示 HASH 的 OID 不加入到公钥加密中。在签名时，OID 总会加到签名值中。

pbSignature：输出值，存放签名后的结果字节串。

pdwSigLen：输出值，存放签名值的长度。

说明：计算数据签名值。

12.1.3.2 示例程序

私钥存放在 CSP 的容器中，在使用私钥前，需要先确定私钥的存放位置，然后打开 CSP 句柄。在获得私钥句柄后，就可以进行签名操作。

确定私钥的位置有两种方式，一是通过查找到的证书定位私钥，二是通过密钥容器定位私钥。应用系统一般使用证书定位私钥方式，CA 系统一般通过容器方式定位私钥。

1. 通过证书句柄获得私钥句柄

```
HCRYPTPROV hCryptProv;
HCRYPTKEY hPrivateKey;
BOOL bFreeProv;
DWORD dwKeySpec;
PCCERT_CONTEXT pCertContext = ...; // 已完成赋值，如通过参数方式获得
// 第一步，通过证书句柄获得 CSP 提供者信息
BOOL rc = CryptAcquireCertificatePrivateKey(
    pCertContext, CRYPT_ACQUIRE_USE_PROV_INFO_FLAG,
    NULL, // reserved
    &hCryptProv, &dwKeySpec, &bFreeProv);
if(!rc) {
    // 处理错误，handleError("CryptAcquireCertificatePrivateKey");
    return rc;
}
```

2. 私钥签名

```
BYTE *data=..., sig[256]; //设置初始数据
DWORD dlen, buflen, dwKeySpec = AT_SIGNATURE;
HCRYPTPROV *hCryptProv
HCRYPTHASH hHash = 0;
BOOL rc = FALSE;
DWORD siglen = buflen;
// 第一步，使用 MD5 算法创建哈希对象
rc = CryptCreateHash(hCryptProv, CALG_MD5, 0, 0, &hHash);
if(!rc) return 0;
// 第二步，计算签名原文数据 HASH 值
rc = CryptHashData(hHash, data, dlen, 0);
if(!rc) {
    CryptDestroyHash(hHash); // 如果出现错误，释放哈希对象句柄
    return 0;
}
// 因为是签名，使用私钥类型 AT_SIGNATURE;在 CryptSignHash()中，如果没有置 CRYPT_
// NOHASHOID 标记，自动在 HASH 结果前加 OID 后再使用私钥加密
// 第三步，执行签名操作
rc = CryptSignHash(hHash, dwKeySpec, NULL, 0, sig, &siglen);
```

```

// 释放哈希对象句柄
CryptDestroyHash(hHash);
if (!rc) return 0;
// 第四步，需要对签名结果字节流进行反序，以便与其他密码库签名结果互通
// reverseBuffer(sig, siglen);
return siglen;

```

12.2 PKCS#11

12.2.1 PKCS#11 简介

PKCS#11 是 RSA 公司定义的安全编程接口，其目的是屏蔽密码设备的差异性，向应用层提供一套统一的编程接口。现有很多密码设备提供了 PKCS#11 编程接口，包括密码服务器、智能密码密钥、密码卡、IC 卡等硬件设备。当然也可以使用软件实现，如火狐浏览器（firefox）内嵌了 PKCS#11 安全模块，用于 SSL 通道建立和密码运算。

12.2.1.1 Cryptoki

PKCS#11 编程接口称作 Cryptoki，是 cryptographic token interface（密码令牌接口）的缩写。它遵循一种基于对象的简单方法，提出技术独立（支持各种各样的设备）和资源共享（多个应用程序可访问多个设备）的目标，应用程序以逻辑视图方式把所有密码设备当作一种密码令牌。应用层不关心密码令牌是如何实现的，它只要调用标准的接口，即可完成密码运算。

Cryptoki 的通用模型如图 12-2 所示。

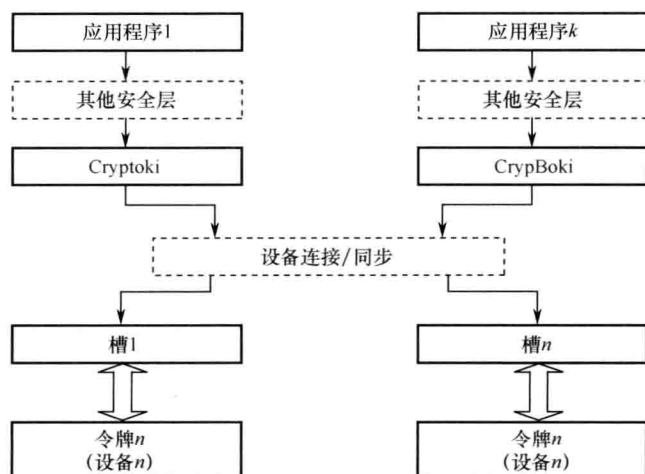


图 12-2 通用 Cryptoki 模型

在系统中，密码设备通过槽（slot）进行运行，这些槽可能是一个物理读卡器，Cryptoki 为这些设备提供操作接口，若一台密码设备存在于读卡器中，一个令牌就存在于该槽中。由于 Cryptoki 提供槽和令牌的逻辑视图，所以槽和令牌可能有其他的物理交互。多个槽可能共享一个读卡器，一个系统可以有多个槽，应用程序能连接到这些槽的任何一个或全部

槽的令牌上。

密码设备可以按照某一命令集执行某些密码操作，这些命令通常要经过标准设备驱动程序，例如 PCMCIA 卡服务程序或槽服务程序。Cryptoki 使每个密码设备看起来逻辑上很像其他设备，而不管是用什么技术实现的。因此，应用程序不必直接与设备驱动器接口（或甚至不必知道包括哪些设备）交互，Cryptoki 隐藏了这些细节。另外，基础设备完全能用软件来实现（例如，在一个服务器上运行的处理程序），不需要专用硬件。

Cryptoki 可以作为支持接口功能的库来实现，而应用程序则与该库连接。应用程序可以直接与 Cryptoki 连接，或者 Cryptoki 是一个“共享”库（或动态连接库），在这种情况下，应用程序动态地连接库。为了防止动态库被替换掉而带来泄密风险，从安全角度来说，一般建议直接连接该库。

设备的种类和所支持的能力的种类将取决于专用 Cryptoki 库。PKCS#11 标准只定义库的接口，不定义库的特征。要特别指出的是，并不是所有的库都支持这个接口（因为不是所有的令牌支持所有的机制）中定义的机制（算法），并且库也许只支持可使用的所有密码设备的一个子集。

12.2.1.2 令牌逻辑视图

Cryptoki 的令牌逻辑视图是一个能存储对象和能执行密码函数的设备。Cryptoki 定义了 3 个对象：数据、证书和密钥。数据对象由应用程序定义。一个证书对象存储一个证书，一个密钥对象存储一个密钥。密钥可以是一个公开密钥、一个私有密钥或是一个对称密钥，每个种类的密钥在专用机制中使用其定义的辅助类型。令牌的这种逻辑视图如图 12-3 所示。

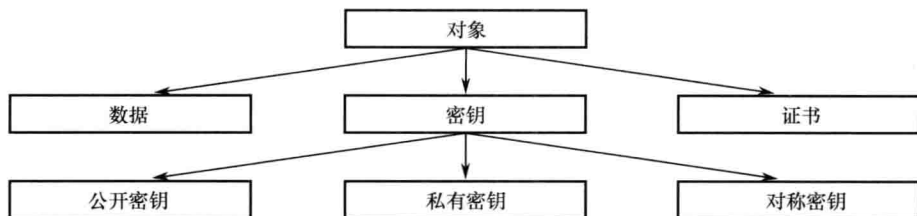


图 12-3 令牌逻辑视图

对象根据它们的使用期限和可见度进行分类。“令牌对象”能被所有与之连接并有足够权限的应用程序访问。即使关闭应用程序与令牌之间的“会话”和令牌从其槽中移除后，“令牌对象”仍保存在令牌上。而“会话对象”更具有暂时性，当以任何一种方式关闭会话时，由该会话产生的所有会话对象就会自动被销毁。此外，会话对象只能由产生它们的应用程序访问。

12.2.1.3 Cryptoki 的几种常用概念

为了便于理解，下面归纳介绍了 Cryptoki 几种常用的概念。

（1）槽

槽是一个逻辑读卡器，若一台密码设备存在于读卡器中，就说一个令牌存在于该槽中。

（2）令牌

令牌是一个能存储对象、执行密码功能的设备。

(3) 会话

会话是应用程序和令牌之间的逻辑连接。打开一个会话后，应用程序便访问令牌的公共对象。概括来讲，一个打开的会话能用来执行 3 类操作：管理操作（如注册）、对象管理操作（如在令牌上建立或销毁一个对象）和密码操作（如计算一个消息摘要）。

(4) 对象

对象是属性的集合，对象可根据生命周期分为会话对象和令牌对象。应用程序打开一个会话时，Cryptoki 以会话句柄方式标识会话，便于应用程序使用会话句柄访问会话，同时应用程序可通过调用 Cryptoki 接口创建或查询对象，而 Cryptoki 以对象句柄方式标识对象，便于会话通过对象句柄访问对象。

(5) 密钥

密钥可以是一个公钥、一个私钥或一个对称密钥。

(6) 机制

机制详细说明某种加密过程是如何执行的。一个机制作为一个对象，明确指定了一个密码处理是如何执行的。Cryptoki 中定义的机制被不同的密码操作支持。对于一个特定的令牌，一个特定的操作只支持 Cryptoki 中定义的机制集合的一个子集。

(7) 属性

属性描述了对象的特征。在创建对象时，Cryptoki 对哪些属性必须指定、哪些可选都有明确的规定，并且明确了属性之间的一致性定义。在对象复制时，能否修改对象的属性依赖于确定对象的属性值，如一个保密密钥对象在使用 C_CopyObject 进行对象复制时其 CKA_SENSITIVE 属性的值可以由 CK_FALSE 改变为 CK_TRUE，而不能由 CK_TRUE 改变为 CK_FALSE。

(8) 用户类型

Cryptoki 识别 3 种用户类型：管理员、普通用户、上下文特定用户。只有普通用户通过认证之后才能访问令牌上的私有对象。

管理员的角色是初始化一个令牌，并设置普通用户的 PIN（口令），或许还要操作一些公用对象。普通用户在管理员没有为其设置 PIN 之前是不能登录令牌的。事实上，管理员和普通用户可以是一个人，这可根据密码令牌的管理策略来确定。上下文特定用户只有在重新认证密码操作时才会用到。

(9) 会话句柄和对象句柄

一个会话句柄是一个 Cryptoki 赋予的会话值。在许多方式中它类似于一个文件句柄。对于函数来说，它被规定用来确定函数应当执行哪一个会话。一个应用程序的所有线程能平等地访问所有的会话句柄。

对象句柄是用来控制 Cryptoki 对象的标识符。与会话句柄类似，一个对象句柄的可见性在一个应用程序的所有线程中都是相同的。只读会话仅能对令牌对象进行只读访问，读写会话对令牌对象能进行读/写访问。

12.2.1.4 Cryptoki 前缀说明

为了便于标记，Cryptoki 对函数、类型定义、常量、变量都增加了标识前缀，如表 12-2 所示。

表 12-2 Cryptoki 前缀说明

前 缀	说 明	前 缀	说 明
C_	函数	CKP_	伪随机功能
CK_	数据类型或通用常数	CKS_	会话状态
CKA_	属性	CKR_	返回值
CKC_	证书类型	CKU_	用户种类
CKF_	位标志	CKZ_	Salt/Encoding 参数资源
CKG_	屏蔽生成功能	H	句柄
CKH_	硬件特征类型	UI	CK_ULONG
CKK_	密钥类型	P	指针
CKM_	机制类型	Pb	指向 CK_BYTE 的指针
CKN_	通知	Ph	指向句柄的指针
CKO_	对象级别	Pul	指向 CK_ULONG 的指针

12.2.1.5 调用 Cryptoki 的一般过程

为了完成一次密码操作，需要经过：①初始化环境；②获取槽列表；③打开会话；④用户身份登录；⑤进行相关操作。完成操作后。需要：⑥登出；⑦关闭会话；⑧清空环境。

1. 初始化环境函数

```
CK_DEFINE_FUNCTION(CK_RV, C_Initialize)(
    CK_VOID_PTR pInitArgs);
```

C_Initialize 初始化 Cryptoki 库。pInitArgs 取值为 NULL_PTR，或为指向含有说明库应该如何处理多线程访问信息的 CK_C_INITIALIZE_ARGS 结构。

2. 获取槽列表函数

```
CK_DEFINE_FUNCTION(CK_RV, C_GetSlotList)(
    CK_BBOOL tokenPresent,
    CK_SLOT_ID_PTR pSlotList,
    CK_ULONG_PTR pulCount);
```

C_GetSlotList 用于获得系统中的一个槽列表。tokenPresent 表明所获得的列表是否只包括带有一个当前令牌（TRUE）的那些槽，或者包括所有槽（FALSE）；pulCount 指向接收槽数量的单元。

应用程序调用 C_GetSlotList 有两种方法：

① 如果 pSlotList 是 NULL_PTR，那么 C_GetSlotList 所能做的一切是返回（in*pulCount）槽的数量，而不是实际返回一个槽列表。在 C_GetSlotList 入口上由 pulCount 指向的缓冲区的内容此时无意义，该调用返回值 CKR_OK。

② 如果 pSlotList 不是 NULL_PTR，那么*pulCount 必须包含由 pSlotList 所指的缓冲区的大小（以 CK_SLOT_ID 单元为单位）。如果该缓冲区有足够大的空间容纳槽列表，那么在该缓冲区中返回该列表，并返回 CKR_OK。反之，调用 C_GetSlotList 返回值 CKR_BUFFER_TOO_SMALL。在任何一种情况下，值*pulCount 被设置为可容纳槽数的大小。

由于 C_GetSlotList 不分配自己的空间，因此应用经常两次调用 C_GetSlotList。然而，多次调用 C_GetSlotList 并不是必需的。

C_GetSlotList 报告的所有槽必须能够被 C_GetSlotInfo 有效查询。另外，通过 Cryptoki 库可访问的槽的设置是在调用 C_Initialize 时就被确定。如果一个应用调用 C_Initialize 和 C_GetSlotList，那么用户接入一个新的设备，如果再次调用 C_GetSlotList 的话，该设备不能突然一下子作为一个新槽出现。为了识别新的设备，需要首先调用 C_Initialize（为了能够成功调用 C_Initialize，需要首先调用 C_Finalize）。即使 C_Initialize 被成功调用，新设备也有可能不能被成功识别。在某些平台上，可能需要重新启动整个系统。

3. 打开会话函数

```
CK_DEFINE_FUNCTION(CK_RV, C_OpenSession)(
    CK_SLOT_ID slotID,
    CK_FLAGS flags,
    CK_VOID_PTR pApplication,
    CK_NOTIFY iNotify,
    CK_SESSION_HANDLE_PTR phSession);
```

C_OpenSession 打开某一特定槽中应用和令牌间的会话。slotID 是槽的 ID；flags 表明会话的类型；pApplication 是一个要传递给通知回调的应用定义的指针；iNotify 是通知回调函数的地址；phSession 指向接收新会话句柄的单元。

当打开一个带 C_OpenSession 的会话时，标志参数由 CK_SESSION_INFO 数据类型中定义的零位或多位标志的逻辑或组成。由于传统原因，必须总是设置 CKF_SERIAL_SESSION 位；如果调用 C_OpenSession 时该位没有设置，调用则不成功，返回错误码 CKR_PARALLEL_NOT_SUPPORTED。

一个应用程序可能并行打开的会话数量会有一个限制，它取决于会话是“只读”还是“读/写”。尝试打开一个由于有太多某种类型的现有会话而不能成功时，会返回 CKR_SESSION_COUNT。

如果令牌是写保护的（如 CK_TOKEN_INFO 结构中所说），那么可以打开带令牌的只读会话。

如果调用 C_OpenSession 的应用程序已经有带令牌的管理员打开的读/写会话，那么任何试图打开带令牌的只读会话均失败，返回错误码 CKR_SESSION_READ_WRITE_SO_EXISTS。

Cryptoki 采用 iNotify 回调函数来指明某些事件的应用。如果应用不希望支持回调，则应该传输一个 NULL_PTR 值作为 iNotify 参数。

4. 用户身份登录函数

```
CK_DEFINE_FUNCTION(CK_RV, C_Login)(
    CK_SESSION_HANDLE hSession,
    CK_USER_TYPE userType,
    CK_CHAR_PTR pPin,
    CK_ULONG ulPinLen);
```

C_Login 为用户注册一个令牌。hSession 是会话句柄；userType 是用户类型；pPin 指向用户的 PIN；ulPinLen 是 PIN 的长度。

如果调用成功，则根据用户的类型，应用的各个会话要么进入“读/写管理员函数”状态，要么进入“只读用户函数”状态。

如果正如 CK_TOKEN_INFO 中的 CKF_PROTECTED_AUTHENTICATION_PATH 标志所表明的，令牌有一条“受保护的认证路径”，那就意味着应用不需要通过 Cryptoki 库发送 PIN，用户也有某种办法能让其被令牌认证。其中一种可能的方法是用户进入令牌自身的或槽设备的 PIN 键盘上的 PIN。或者用户可能甚至不使用 PIN——比如说，可以由某种指纹阅读设备取得认证。为了注册一个带受保护认证的路径，C_Login 的 pPin 参数应该是 NULL_PTR。当 C_Login 返回时，令牌支持的任何方法都将被执行；返回值 CKR_OK 说明用户已被成功认证，返回值 CKR_PIN_INCORRECT 则说明用户被拒绝访问。

5. 登出函数

```
CK_DEFINE_FUNCTION(CK_RV, C_Logout)(  
    CK_SESSION_HANDLE hSession);
```

C_Logout 将用户从令牌中注销。hSession 是会话的句柄。

如果调用成功，应用的每个会话根据当前的用户类型或者进入“读/写公共会话”状态，或者进入“只读公共会话”状态。

当 C_Logout 成功执行后，专用对象的任何应用的句柄就都无效了（即使用户以后又返回注册到令牌，那些应用仍然是无效的）。另外，属于应用的会话的所有专用会话对象都被破坏。

如果应用程序的会话中有任何活动密码操作或对象查找的操作，而且接着该应用成功执行 C_Logout，那么那些操作也有可能不再是有效的。因此，注销前应当结束任何正在执行的操作。

6. 关闭会话

```
CK_DEFINE_FUNCTION(CK_RV, C_CloseSession)(  
    CK_SESSION_HANDLE hSession);
```

C_CloseSession 关闭应用和令牌间的会话。hSession 是会话的句柄。

当关闭一个会话时，会话创建的所有会话对象均被自动销毁，即使应用还有其他会话在“使用”该对象。

如果该函数执行成功，关闭了应用程序和令牌间的最后一个会话，那么应用令牌的逻辑状态返回到公共会话。应用程序打开令牌的任何新的会话或者是只读公共会话或者是读/写公共会话。

尽管假设该 C_CloseSession 关闭一个会话，返回值 CKR_SESSION_CLOSED 是一个错误的返回。它事实上表明（有时不可能）在该函数正在执行时，应用程序又调用 C_CloseSession 来关闭该会话，并且该调用首先结束执行。

7. 清空环境函数

```
CK_DEFINE_FUNCTION(CK_RV, C_Finalize)(  
    CK_VOID_PTR pReserved);
```

调用 C_Finalize 表明，用 Cryptoki 库完成了一个应用程序。它应当是应用程序所做的最后一次调用。pReserved 参数为未来版本使用。对于该版本，它应当被设置成 NULL_PTR。

如果几个应用程序正在使用 Cryptoki，则每个应用程序应当调用 C_Finalize。

12.2.2 使用证书

12.2.2.1 函数说明

1. C_FindObjectsInit 函数

```
CK_DEFINE_FUNCTION(CK_RV, C_FindObjectsInit)(
    CK_SESSION_HANDLE hSession,
    CK_ATTRIBUTE_PTR pTemplate,
    CK_ULONG ulCount);
```

C_FindObjectsInit 按匹配模板，初始化对令牌和会话中的对象的查找。hSession 是会话的句柄；pTemplate 指向确定要匹配的属性值的查找模板；ulCount 是查找模板中的属性数。匹配标准是与模板中所有属性精确的逐字节的匹配。为了找到所有的目标，可将 ulCount 设置为 0。

调用 C_FindObjectsInit 后，应用程序可能一次或多次调用 C_FindObjects 以获得匹配模板的对象的句柄，接着最终调用 C_FindObjectsFinal 来结束活动的查找操作。在一个给定的会话中一次至多有一个查找操作是活动的。

查找只能发现会话能看见的对象。比如，“读/写公共会话”中的对象查找不能发现任何专用对象（即使查找模块中的一个属性指定该查找用于专用对象）。

2. C_FindObjects 函数

```
CK_DEFINE_FUNCTION(CK_RV, C_FindObjects)(
    CK_SESSION_HANDLE hSession,
    CK_OBJECT_HANDLE_PTR phObject,
    CK_ULONG ulMaxObjectCount,
    CK_ULONG_PTR pulObjectCount);
```

C_FindObjects 继续查找匹配模板的令牌和会话中的对象，获得对象句柄。hSession 是会话句柄；phObject 指向接收对象句柄列表（数组）的单元；ulMaxObjectCount 是要返回的对象句柄的最大数；pulObjectCount 指向接收实际返回对象句柄数的单元。

如果已没有匹配模板的对象，那么 pulObjectCount 所指单元接收值为 0。调用 C_FindObjects 前必须先调用 C_FindObjectsInit 成功。

3. C_FindObjectsFinal

```
CK_DEFINE_FUNCTION(CK_RV, C_FindObjectsFinal)(
    CK_SESSION_HANDLE hSession);
```

C_FindObjectsFinal 结束查找令牌和会话对象。hSession 是会话的句柄。

4. C_GetAttributeValue 函数

```
CK_DEFINE_FUNCTION(CK_RV, C_GetAttributeValue)(
    CK_SESSION_HANDLE hSession,
    CK_OBJECT_HANDLE hObject,
```



```
CK_ATTRIBUTE_PTR pTemplate,
CK_ULONG ulCount);
```

C_GetAttributeValue 获得对象的一个或多个属性值。hSession 是会话的句柄；hObject 是对象的句柄；pTemplate 指向规定即将获得属性值的模板，并接收属性值；pulCount 是模板中的属性数。

对于属性模板中的每个三元组（类型，值，长度），C_GetAttributeValue 执行以下的算法。

① 如果由于对象敏感或不可获取因而不能返回对象的规定属性（即类型字段规定的属性）的话，那么该三元组中的 ulValueLen 字段被修改，包含的值为-1（即当转换为 CK_LONG 类型时，包含值-1）。

② 否则，如果对象的规定属性无效（对象没有这种属性），那么该三元组中的 ulValueLen 字段被修改，包含的值为-1。

③ 否则，如果 pValue 字段的值为 NULL_PTR，那么 ulValueLen 字段被修改，包含的是对象指定属性的实际长度。

④ 否则，如果 ulValueLen 中规定的长度足够长，能包含对象指定属性的值，那么属性就被复制到 pValue 中的缓冲区里，ulValueLen 字段被修改，包含属性的实际长度。

⑤ 否则，ulValueLen 字段被修改，包含值-1。

如果情况①应用于任何请求的属性，那么调用返回 CKR_ATTRIBUTE_SENSITIVE 值。如果情况②应用于任何请求的属性，那么调用返回 CKR_ATTRIBUTE_TYPE_INVALID 值。如果情况⑤应用于任何请求的属性，那么调用返回 CKR_BUFFER_TOO_SMALL 值。通常，如果这些错误码中的不止一个能用，Cryptoki 可能返回其中的任意一个。只有如果没有一个能适用于请求的属性时才返回 CKR_OK。

12.2.2.2 示例程序

完成证书访问，需要进行如下操作步骤：①初始化环境；②获取槽列表；③打开会话；④用户身份登录；⑤按证书属性查找；⑥登出；⑦关闭会话；⑧清空环境。

1. Cryptoki 通用调用过程代码

```
CK_SLOT_ID      slotList[32];
CK_ULONG        ulSlotCount = 32;
CK_RV           rv = CKR_OK;
CK_SESSION_HANDLE hSession = 0;
char             *passwd=" 111111" ;
int             passlen = 6;
// 初始化环境
rv = C_Initialize(NULL_PTR);
if (rv != CKR_OK) { return BAD_INIT_ERROR; }
// 获取槽列表
rv = C_GetSlotList(TRUE, &slotList[0], &ulSlotCount);
if (rv != CKR_OK) { return BAD_SLOT_ERROR; }
// 打开会话
```

```

rv = C_OpenSession(slotList[0],
    CKF_SERIAL_SESSION | CKF_RW_SESSION,
    NULL, NULL, &hSession);
if (rv != CKR_OK) { return BAD_SESSION_ERROR; }
// 用户身份登录
rv = C_Login(hSession, CKU_USER, passwd, passlen);
if (rv != CKR_OK) { return BAD_LOGIN_ERROR; }
// 具体业务操作
:
// 登出
rv = C_Logout(hsession);
// 关闭会话
rv = C_CloseSession(hsession);
// 清空环境
rv = C_Finalize(NULL);

```

2. 查询证书过程代码

```

#define numof(arr) (sizeof(arr)/sizeof((arr)[0]))
CK_SESSION_HANDLE session = ...; // 假定会话已经打开，并成功登录
int rv, res = -1;
CK_OBJECT_HANDLE obj;
CK_ULONG count;

CK_OBJECT_CLASS cert_search_class = CKO_CERTIFICATE;
CK_ATTRIBUTE cert_search_attrs[] = {
    {CKA_CLASS, &cert_search_class, sizeof(cert_search_class)} };
// 初始化查找对象操作
rv = C_FindObjectsInit(session, cert_search_attrs, numof(cert_search_attrs));
if (rv != CKR_OK) { return BAD_FIND_INIT_ERROR; }
// 循环查找匹配的证书对象
do {
    rv = C_FindObjects(session, &obj, 1, &count);
    // 获得证书 DER 编码值
    res = get_cert_data(session, obj);
    // 如果找到，继续下一个证书对象
} while (rv == CKR_OK && count == 1);
// 完成查找，关闭查找对象
rv = C_FindObjectsFinal(session);

```

3. 获取证书 DER 编码值代码

证书数据就在证书对象的属性里，需要获取证书对象的 CKA_VALUE 属性。

```

int get_cert_data(CK_SESSION_HANDLE session, CK_OBJECT_HANDLE obj)
{

```

```

char cert_data[2048];
int cert_len;
CK_ATTRIBUTE templ;
int rv;
templ.type = CKA_VALUE;
templ.pValue = cert_data;
templ.ulValueLen = sizeof(cert_data);
rv = C_GetAttributeValue(session, o, &templ, 1);
if (rv != CKR_OK) {    return -1;    }
cert_len = templ.ulValueLen;
// cert_data 中已经保存了证书的 DER 值
return 0;
}

```

12.2.3 使用私钥

12.2.3.1 函数说明

1. C_SignInit 函数

签名的第一步是获得私钥对象。通过私钥的标签值，调用 FindObjectsInit、FindObjects 和 FindObjectsFinal，找到私钥对象，接着利用私钥对象调用 C_SignInit 和 C_Sign 完成签名。这两个函数的说明如下。

```

CK_DEFINE_FUNCTION(CK_RV, C_SignInit)(
    CK_SESSION_HANDLE hSession,
    CK_MECHANISM_PTR pMechanism,
    CK_OBJECT_HANDLE hKey);

```

C_SignInit 初始化签名操作。hSession 是会话的句柄；pMechanism 指向签名机制；hKey 是签名密钥的句柄。

调用 C_SignInit 后，可直接调用 C_Sign 对单部分数据进行签名，或者先调用 C_SignUpdate 一次或多次，接着调用 C_SignFinal 给多部分中的数据签名。签名操作是有效的，直到应用程序调用 C_Sign 或 C_SignFinal 真正获得签名。要处理另外的数据（单部分或多部分），应用程序必须再次调用 C_SignInit。

2. C_Sign 函数

```

CK_DEFINE_FUNCTION(CK_RV, C_Sign)(
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pData,
    CK_ULONG ulDataLen,
    CK_BYTE_PTR pSignature,
    CK_ULONG_PTR pulSignatureLen);

```

C_Sign 给单部分中的数据签名。hSession 是会话句柄；pData 指向数据；ulDataLen 是

数据长度；pSignature 指向接收签名的单元；pulSignatureLen 指向包含签名长度的单元。

签名操作必须用 C_SignInit 初始化。调用 C_Sign 总是会结束有效的签名操作，除非它返回 CKR_BUFFER_TOO_SMALL 或这是一次成功的调用（即返回 CKR_OK）来确定包含签名所需的缓冲区的长度。

不能使用 C_Sign 来结束多部分签名操作，C_Sign 必须在不插入 C_SignUpdate 的情况下在 C_SignInit 之后调用。

对于大部分机制，C_Sign 相当于一系列 C_SignUpdate 操作后面跟着 C_SignFinal。

3. C_SignUpdate

```
CK_DEFINE_FUNCTION(CK_RV, C_SignUpdate)(
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pPart,
    CK_ULONG ulPartLen);
```

C_SignUpdate 继续多部分签名操作，处理另一个数据部分。hSession 是会话句柄；pPart 指向数据部分；ulPartLen 是数据部分的长度。

签名操作必须用 C_SignInit 初始化。这一函数可以连续调用若干次。调用 C_SignUpdate 产生错误时终止当前的签名操作。

4. C_SignFinal

```
CK_DEFINE_FUNCTION(CK_RV, C_SignFinal)(
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pSignature,
    CK_ULONG_PTR pulSignatureLen);
```

C_SignFinal 结束多部分签名操作，返回签名值。hSession 是会话句柄；pSignature 指向接收签名的单元；pulSignatureLen 指向包含签名长度的单元。

签名操作必须用 C_SignInit 初始化。调用 C_SignFinal 总是会结束活动的签名操作，除非它返回 CKR_BUFFER_TOO_SMALL 或这是一次成功的调用（即返回 CKR_OK）来确定包含签名所需的缓冲区的长度。

12.2.3.2 示例程序

使用私钥进行签名，需要进行如下操作：①初始化环境；②获取槽列表；③打开会话；④用户身份登录；⑤进行签名操作；⑥登出；⑦关闭会话；⑧清空环境。

除“⑤进行签名操作”外，其他步骤与 12.2.2 节中“Cryptoki 通用调用过程代码”相同。

1. 私钥签名过程

```
// 通过 label 属性获取私钥对象
CK_OBJECT_HANDLE key;
CK_RV rv = CKR_OK;
CK_KEY_TYPE keyType_rsa = CKK_RSA;
CK_OBJECT_CLASS private_key_class = CKO_PRIVATE_KEY;
```

```

CK_BBOOL          sw_false = 0;
CK_SESSION_HANDLE hSession = ...; // 见 C_OpenSession 调用
char              private_key_label[]="private key label";
CK_ULONG          ulObjectCount;
CK_ATTRIBUTE       attrtemplate[1][5] = { {
    {CKA_CLASS,      &private_key_class, sizeof(private_key_class)},
    {CKA_KEY_TYPE,   &keyType_rsa, sizeof(keyType_rsa)},
    {CKA_TOKEN,      &sw_false, sizeof(sw_false)},
    {CKA_PRIVATE,    &sw_false, sizeof(sw_false)},
    {CKA_LABEL,      private_key_label, sizeof(private_key_label)-1}
} };
CK_MECHANISM       rsa_Mechasm = {CKM_SHA1_RSA_PKCS, NULL_PTR, 0};
CK_BYTE            pSignature[1024]; /* gets the signature */
CK_ULONG           ulSignatureLen = sizeof(pSignature); /* gets signature length */
char *data=...;
int dalen = ....;
rv = C_FindObjectsInit(hSession, attrtemplate[0], 5);
if (rv != CKR_OK) { return -2; }
ulObjectCount = 0;
do {
    rv = C_FindObjects(hSession, &key, 1, &ulObjectCount);
    if (rv != CKR_OK) {
        C_FindObjectsFinal(hSession);
        return -3;
    }
} while (ulObjectCount == 1);
rv = C_FindObjectsFinal(hSession);
if (rv != CKR_OK) { return -4; }
// 进行签名操作
rv = C_SignInit(hSession, &rsa_Mechasm, key);
if (rv != CKR_OK) { return -5; }
rv = C_Sign(hSession, data, dalen, pSignature, &ulSignatureLen);
if (rv != CKR_OK) { return -6; }
// C_Sign 完成后, pSignature 存储了签名结果, ulSignatureLen 保存了签名值长度

```

12.3 JCA/JCE

12.3.1 JCA/JCE 简介

JCA (Java Cryptography Architecture, Java 密码体系结构) 和 JCE (Java Cryptography Extension, Java 密码扩展包) 是 Java 平台提供的用于安全加密服务的两组 API, 但它们并不实现任何算法, 它们只是连接应用程序和实际算法实现程序的一组接口。软件开发商可以根据 JCE 接口把各种算法实现后, 打包成一个安全实现提供者, 动态地加载到 Java 运行环境中。

根据美国出口限制的规定, JCA 可出口 (JCA 和 Sun 的一些默认实现包含在 Java 发行版中), 但 JCE 对部分国家是限制出口的。因此, 要实现一个完整的安全功能, 就需要一个或多个第三方厂商提供 JCE 实现产品, 其称为安全提供者。

安全提供者是承担特定安全机制实现的第三方, 有些提供者是完全免费的, 而另一些提供者则需要付费。安全提供者的公司有 Sun、BouncyCastle 等。Sun 提供了如何开发安全提供者的细节。BouncyCastle 提供了可以在 J2ME/ J2EE/J2SE 平台应用的 API, 而且 BouncyCastle 是免费的。

在 JDK 1.4 之前, JCE 并不随 JDK 一起发布, 正因如此, JCA 和 JCE 经常被称为独立的不同组件。随着 JCE 捆绑在 JDK 1.4 以后版本中一起发布, JCA 和 JCE 的区别变得越来越不明显, 由于 JCE 使用与 JCA 相同的架构, JCE 越来越被认为是 JCA 的一个组成部分。

在 JDK 中的 JCA 包括两个软件组件: 密码服务框架的定义和密码服务提供者。密码服务框架包括 `java.security`、`javax.crypto`、`javax.crypto.spec` 和 `javax.crypto.interfaces` 等软件包。密码服务提供者包括 Sun、`SunRsaSign`、`SunJCE` 等。每当提及一个特定的 JCA 提供者时, 总是引用提供者的名称。

1. JCA 体系架构

JCA 遵循如下设计原则: 实现上的独立性和互操作性, 算法上的独立性和可扩展性。基于这两个原则, 采用了如下实现技术。

(1) 实现上的独立性

应用程序不需要自己实现安全算法, 它们只需要在 Java 平台中请求安全服务即可。其中, 安全服务的实现是在提供者中完成的, 而提供者通过标准的接口以插件形式存在于 Java 平台中。因此, 实现上的独立性是通过这种基于提供者的架构而取得的。提供者又称为 CSP (Cryptographic Service Provider, 密码服务提供者)。

(2) 实现上的互操作性

提供者在各种应用程序之间可以彼此协作, 也就是应用程序可以使用不同的提供者, 同时, 一个提供者也可以为多个程序服务。

(3) 算法上的独立性

Java 平台定义了一些不同类型的与密码相关的引擎 (Engines) 或者叫服务 (Services), 然后定义了为该引擎提供功能的类, 叫作引擎类。比如, `MessageDigest`、`Signature`、`KeyFactory`、`Cipher` 等。

(4) 算法上的可扩展性

第三方的密码算法可以通过扩展 JCA 给出的某些类来方便地添加。

JCA 与应用程序的关系如图 12-4 所示, `ProviderB` 和 `ProviderC` 都实现了 MD5 算法。

`java.security.Provider` 是所有这些提供者的基础类。每个提供者包含该密码服务提供者的名字以及它所实现的所有算法的列表。当应用程序需要使用一个特定的算法时, 它可以向 JCA 请求, JCA 框架就会访问所有可用的提供者, 然后从中选中一个合适的提供者, 并为该应用程序创建一个算法的实例。比如:

```
md = MessageDigest.getInstance("MD5");
md = MessageDigest.getInstance("MD5", "ProviderC");
```

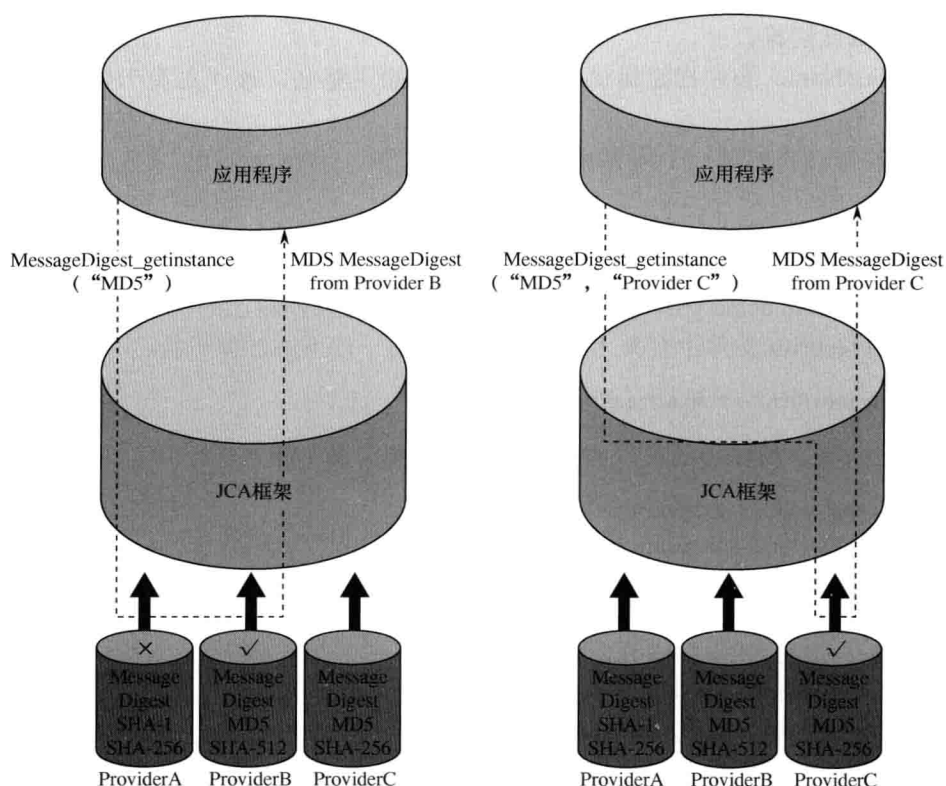


图 12-4 JCA 体系架构

对于上面两句代码，在图 12-4 中有 3 个加密服务提供者，分别是 ProviderA、ProviderB 和 ProviderC，应用程序向 JCA 请求“MD5”算法服务，JCA 通过搜索所有可用的加密提供者来返回合适的算法实现。对于第一句代码，见图 12-4 中左半部分，JCA 查找到 ProviderA，而 ProviderA 提供了“SHA-1”和“SHA-256”算法，并没有提供“MD5”算法，所以不合适，继续搜索，找到 ProviderB，因为它里面有“MD5”算法，故返回该算法的一个实例。

而对于第二句代码，见图 12-4 中右半部分，它指定了从 ProviderC 中获取“MD5”算法服务，故 JCA 直接返回 ProviderC 中“MD5”算法的一个实例。

2. 安装密码服务提供者

首先要使类可用，以便需要时能找到它们。可以按 JAR 文件的形式提供提供者类。把提供者添加到被认可的提供者列表中。这可通过编辑 JDK 的 `lib/security`（在 Windows 中为 `lib\security`）目录下的 `java.security` 文件来完成。对于每个提供者，该文件都应有如下形式的声明：

```
security.provider.n=masterClassName
```

该语句声明了一个提供者并指定了它的优先顺序 `n`。优先顺序是当没有请求特定提供者时，为获得请求算法而查找提供者的顺序。优先顺序是从 1 开始的，1 享有最大优先权，

紧接着是 2，依此类推。

`masterClassName` 必须指定提供者的“主类”的完整名，该“主类”始终是 `Provider` 类的子类。

安装 JDK 时，将包括一个内置提供者，即“SUN”。`java.security` 文件仅提供如下提供者规范：

```
security.provider.1=sun.security.provider.Sun
```

假定主类是 `com.acme.provider` 包中的 `Acme` 类，并且使自己的提供者优先顺序为 2。为此，在 `java.security` 文件中有关“SUN”提供者的一行下添加如下行：

```
security.provider.2 = com.acme.provider.Acme
```

在 JDK 1.6 中，所提供的密码服务提供者的配置信息如下：

```
security.provider.1=sun.security.provider.Sun
security.provider.2=sun.security.rsa.SunRsaSign
security.provider.3=com.sun.net.ssl.internal.ssl.Provider
security.provider.4=com.sun.crypto.provider.SunJCE
security.provider.5=sun.security.jgss.SunProvider
security.provider.6=com.sun.security.sasl.Provider
security.provider.7=org.jcp.xml.dsig.internal.dom.XMLDSigRI
security.provider.8=sun.security.smartcardio.SunPCSC
security.provider.9=sun.security.mscapi.SunMSCAPI
```

3. 引擎（Engine）类和算法

一个“引擎类”以一种抽象方式（无具体的实现方法）定义一种密码服务。密码服务总是与某个特定的算法或类型相关联，它或者提供密码运算（如数字签名或报文摘要的运算），生成或提供密码运算所需要的密码信息（密钥或参数），或者生成安全封装的密钥（这些密钥可用于密码运算中）的数据对象。例如，`Signature` 和 `KeyFactory` 类就是两个引擎类。

引擎类提供了两种类型操作：

- ① 密码操作（加密、数字签名、消息摘要等）；
- ② 生成器或密码参数转换器（密钥和算法参数），或密码对象（如密钥库或证书），密码对象是对加密数据的封装，使密码数据可以在更高抽象层次使用。

JDK 1.6 中定义了常用的引擎类，如表 12-3 所示。

表 12-3 JDK 常用引擎类

引 擎 类	说 明
<code>SecureRandom</code>	用于生成随机数或伪随机数
<code>MessageDigest</code>	用于计算数据的报文摘要
<code>Signature</code>	用于对数据进行签名和校验数字签名
<code>Cipher</code>	用密钥进行初始化，用于加密/解密数据。支持多种类型算法：对称块加密（例如 AES、DES、DESede、Blowfish、IDEA），流加密（如 RC4），非对称加密（例如 RSA），以及基于密码的加密法（PBE）

(续表)

引擎类	说 明
MAC	消息认证码, 如 MessageDigests 一样, 也产生哈希值。它先用密钥进行初始化, 以保护信息的完整性
KeyFactory	用于把类型为 Key 的密钥数据转换为规范密钥, 或进行相反的转换
SecretKeyFactory	用于把类型为 SecretKey 的密钥数据转换成规范密钥, 或进行相反的转换。SecretKey Factorys 是 KeyFactorys 的子类, 用于创建对称密钥
KeyPairGenerator	用于生成与指定算法相匹配的公钥和私钥对
KeyGenerator	用于生成指定算法的新的对称密钥
KeyAgreement	由两方或多方商定, 指定一个密钥进行加密操作
AlgorithmParameters	用于存储某一特定算法的参数, 包括编码和解码参数
AlgorithmParameterGenerator	用于生成一组与指定算法相匹配的参数
KeyStore	用于创建和管理密钥库。密钥库是密钥的数据库。密钥库中的私钥有一个与之关联的证书链, 用于认证对应的公钥。密钥库还含有来自可信实体的证书
CertificateFactory	用于创建公钥证书和证书撤销列表 (CRL)
CertPathBuilder	用于构建证书链 (也称为证书路径)
CertPathValidator	用于验证证书链的有效性
CertStore	用于检索证书和证书撤销列表 (CRL)

注意, 生成器创建全新内容的密码对象, 而工厂是从已有数据 (如一个已经编码好的证书) 创建密码对象。所有的引擎类没有公共构造方法, 必须使用 `getInstance` 方法以获得一个密码对象实例。

12.3.2 使用证书

证书和私钥都存放在密钥库 (KeyStore) 中, 为了演示方便, 先使用 `keytool` 工具产生一个测试密钥库, 然后使用代码访问产生的测试密钥库。

12.3.2.1 使用 keytool 工具产生测试密钥库

1. keytool 工具说明

`keytool` 是一个密钥和证书管理实用程序。它使用户能够管理自己的公私钥对及相关证书, 或数据完整性和认证服务, 以及使用数字签名。它还允许用户缓存与其通信的另一方的公钥 (以证书形式)。该工具还允许用户管理在对称加密/解密 (如 DES) 中使用的秘密密钥。`keytool` 常用命令参数如表 12-4 所示。

表 12-4 JDK keytool 常用命令参数

参 数	说 明
-genkey	在用户主目录中创建一个默认文件 “.keystore”, 并会产生一个 mykey 的别名, mykey 中包含用户的公钥、私钥和证书 (在没有指定生成位置的情况下, keystore 会存在于用户系统默认目录中)
-alias	产生别名。缺省值为 “mykey”
-keystore	指定密钥库的名称 (产生的各类信息将不在.keystore 文件中)
-keyalg	指定密钥的算法 (如 RSA、DSA, 如果不指定则默认采用 DSA)

(续表)

参 数	说 明
-validity	指定创建的证书有效期为多少天，缺省为 90
-keysize	指定密钥长度，缺省为 1024
-storepass	指定密钥库的密码（获取 keystore 信息所需的密码）
-keypass	指定别名条目的密码（私钥的密码）
-dname	指定证书拥有者信息，例如： "CN=名字与姓氏,OU=组织单位名称,O=组织名称,L=城市或区域名称,ST=州或省份名称,C=两字母国家代码"
-list	显示密钥库中的证书信息
-v	显示密钥库中的证书详细信息
-export	将别名指定的证书导出到文件，如 “keytool -export -alias 别名 -keystore 密钥库 -file 证书文件 -storepass 密码”
-file	参数指定导出到文件的文件名，若不指定，读时为标准输入，写时为标准输出
-delete	删除密钥库中的某条目，如 “keytool -delete -alias 别名 -keystore 密钥库 -storepass 密码”
-printcert	查看导出的证书信息，如 “keytool -printcert -file me.crt”
-keypasswd	修改密钥库中指定条目口令。如 “keytool -keypasswd -alias 别名-keypass 旧密码-new 新密码-storepass 密码-keystore 密钥库”
-storepasswd	修改 keystore 口令。如 “keytool -storepasswd -keystore 密钥库-storepass 原始密码-new 新密码”
-import	将已签名数字证书导入密钥库。如 “keytool -import -alias 别名-keystore 密钥库-file 需导入的证书”

2. 生成 keystore

在当前目录产生密钥库 mytest.keystore，执行如下命令：

```
keytool -genkey -alias mytest -keypass test123 -keyalg RSA -keysize 1024 -validity 365 -keystore mytest.keystore -storepass pass123 -dname "CN=mytest, OU=dev, O=bjca, L=haidian, ST=beijing, C=CN"
```

显示已经产生的密钥库内容，执行如下命令：

```
keytool -list -v -keystore mytest.keystore -storepass pass123
```

显示内容为：

Keystore 类型: JKS
Keystore 提供者: SUN
别名名称: mytest
创建日期: 2014-2-12
项类型: PrivateKeyEntry
认证链长度: 1
认证[1]:
所有者: CN=mytest, OU=dev, O=bjca, L=haidian, ST=beijing, C=CN
签发人: CN=mytest, OU=dev, O=bjca, L=haidian, ST=beijing, C=CN
序列号: 52fb3b75
有效期: Wed Feb 12 17:14:29 CST 2014 至 Thu Feb 12 17:14:29 CST 2015
证书指纹:
MD5:2B:83:F8:44:96:C4:5B:75:47:C0:E4:D0:47:96:88:23
SHA1:BD:94:1A:59:0B:A3:F2:33:52:E1:4C:FA:41:0F:D6:56:30:F7:42:84
签名算法名称: SHA1withRSA
版本: 3

在产生密钥库的过程中，产生了一个 RSA 算法的自签名证书，其密钥长度为 1024 位，使用 SHA1withRSA 签名算法。

当然，可以不使用自签名证书，而是产生 RSA 密钥，然后产生证书请求，把证书请求提交到 CA 发证机构签发证书，然后把签发后的证书导入密钥库。

12.3.2.2 读取密钥库中的证书数据代码

在密钥库建立完成后，就可以用代码读取其中的证书了。

```
//需要包含的包
import java.security.*;
import java.io.*;
import java.util.*;
import java.security.cert.*;
import sun.security.x509.*
import java.security.cert.Certificate;
import java.security.cert.CertificateFactory;
//从密钥库中直接读取证书
String pass="pass123";
String alias="mytest";
FileInputStream in=new FileInputStream("mytest.keystore");
KeyStore ks=KeyStore.getInstance("JKS");
ks.load(in,pass.toCharArray());
java.security.cert.Certificate c=ks.getCertificate(alias);
//显示证书指定信息
System.out.println("输出证书信息:\n"+c.toString());
System.out.println("版本号:"+t.getVersion());
System.out.println("序列号:"+t.getSerialNumber().toString(16));
System.out.println("使用者: "+t.getSubjectDN());
System.out.println("签发者: "+t.getIssuerDN());
System.out.println("有效期: "+t.getNotBefore());
System.out.println("签名算法: "+t.getSigAlgName());
```

12.3.3 使用私钥

使用私钥进行签名的示例代码如下：

```
//需要包含的包
import java.security.*;
import java.io.*;
import java.util.*;
import java.security.*;
import java.security.cert.*;
import sun.security.x509.*
```

```

import java.security.cert.Certificate;
import java.security.cert.CertificateFactory;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.Signature;
import java.security.SignatureException;
//从密钥库中读取私钥
String pass="pass123";
String alias= "mytest";
String keypass= "test123";
FileInputStream in=new FileInputStream("mytest.keystore");
KeyStore ks=KeyStore.getInstance("JKS");
ks.load(in,pass.toCharArray());
PrivateKey prk=(PrivateKey)ks.getKey(alias, keypass.toCharArray());
Signature rsa = Signature.getInstance("SHA1withRSA");
rsa.initSign(prk);
// Update and sign the data
Byte[] data="this is to be signed text".getBytes();
rsa.update(data);
byte[] sig = rsa.sign();

```

12.4 CNG

12.4.1 CNG 简介

Windows Vista 引入了新的加密 API 以替代旧的 CryptoAPI，旧的 CryptoAPI 存在于早期版本的 Windows NT 系列和 Windows 95。下一代加密技术（CNG）旨在长期替代 CryptoAPI，取代 CryptoAPI 提供的所有加密基元或服务。CNG 支持 CryptoAPI 提供的所有算法，而且应用更广泛并且包括许多新算法和更灵活的设计，从而为开发人员提供了对如何执行加密操作，以及算法如何协同工作以执行各种操作的更强的控制能力。

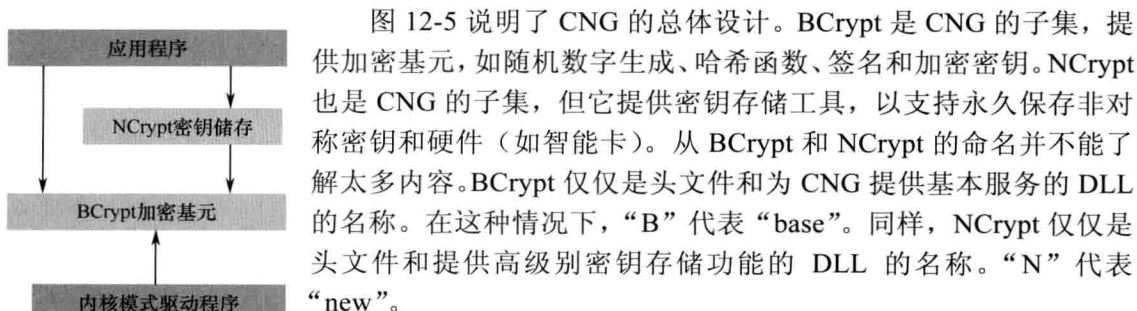


图 12-5 CNG 体系结构

图 12-5 说明了 CNG 的总体设计。BCrypt 是 CNG 的子集，提供加密基元，如随机数字生成、哈希函数、签名和加密密钥。NCrypt 也是 CNG 的子集，但它提供密钥存储工具，以支持永久保存非对称密钥和硬件（如智能卡）。从 BCrypt 和 NCrypt 的命名并不能了解太多内容。BCrypt 仅仅是头文件和为 CNG 提供基本服务的 DLL 的名称。在这种情况下，“B”代表“base”。同样，NCrypt 仅仅是头文件和提供高级别密钥存储功能的 DLL 的名称。“N”代表“new”。

BCrypt 提供的加密基元可在内核模式下直接使用，第一次为用户模式和内核模式应用程序提供公用加密框架，而 NCrypt 提供的密钥存储工具只能用于用户模式应用程序。

有两种主要方法可查看 CNG 提供的加密基元。第一种方法是作为一组逻辑对象，这些对象提供可以调用的方法和可以查询（有时可以修改）的属性。这些对象包括算法提供程序、哈希函数、密钥和密码协议。随机数生成、签名和不同类型的密钥来自哪里？实际上，随机数生成由算法提供程序直接处理，而密钥提供几乎所有其他项。它们可以表示对称或非对称密钥，并用于签发和验证哈希签名。哈希对象和密钥对象从算法提供程序派生而来，而密码协议从一对密钥对象（希望安全地进行通信的一个主体的公钥和另一个主体的私钥）派生而来。

查看 CNG 的另一种方法是作为软件路由器或加密操作的中介。CNG API 基于一组逻辑加密接口构建。例如，可以使用哈希接口，而无需对特定哈希算法实现进行硬编码。大多数加密 API 采用以算法为中心的方法，而 CNG 采用以接口为中心的方法。这为开发人员和应用程序管理员提供了更大的灵活性，在发布的应用程序中使用的算法有缺陷时，他们可以替换该算法。图 12-6 说明了这种以接口为中心的 CNG 视图。

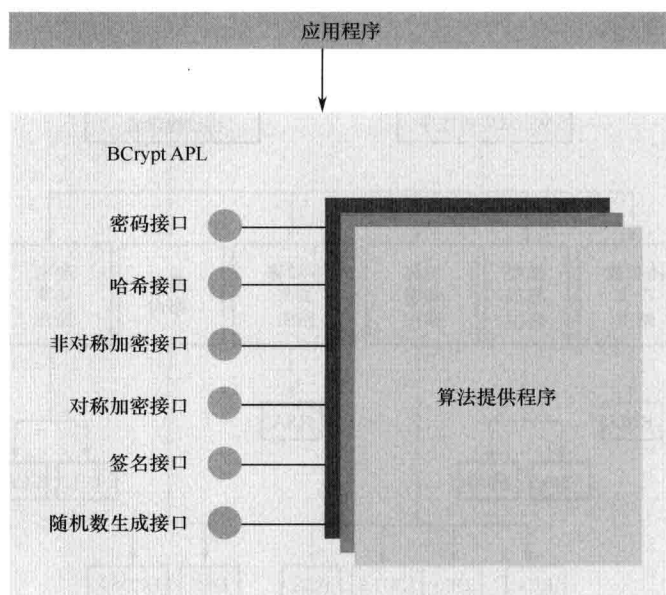


图 12-6 CNG 接口

CNG 允许开发人员不必指定算法提供程序即可请求算法，提供程序负责算法实现，而正是 CNG 的这一方面允许管理员重新配置应用程序以使用不同的实现，无论在应用程序中以声明的方式使用，还是在整个系统策略中使用。

1. CNG 功能特点

① CNG 允许客户使用各自的加密算法或执行标准加密算法。客户还可以添加新的算法。

② CNG 支持采用内核模式加密。为了完全支持加密功能，同时在内核模式和用户模式下使用相同的 API。除了使用 CNG 的启动过程以外，SSL/TLS 与 IPsec 也在内核模式下操作。

- ③ CNG 已通过 FIPS 140-2 2 级认证。
- ④ CNG 在安全过程中使用并存储长效密钥。
- ⑤ CNG 支持当前的 CryptoAPI 1.0 算法。
- ⑥ CNG 支持椭圆曲线加密 (ECC) 算法。
- ⑦ 任何具备受信任的平台模块 (TPM) 的计算机均可在 TPM 中提供密钥隔离和密钥存储。

2. CNG 加密基元结构

CNG 封装多种加密算法。每一种算法或算法类输出基元 API。一个给定的算法的多种实现可以同时共存，但是一个算法只有一个默认实现。

CNG 中每个算法类可以表示为基元路由。在用户模式下使用基元函数的应用程序链接到 BCrypt.dll 二进制路由文件，在内核模式的应用程序链接到 Ksecdd.sys 路由文件。各种路由器组件管理所有的算法基元。这些路由跟踪系统安装的每个算法实现，以及跟踪算法基元实现者的函数调用。

图 12-7 说明了 CNG 加密基元的结构和功能。

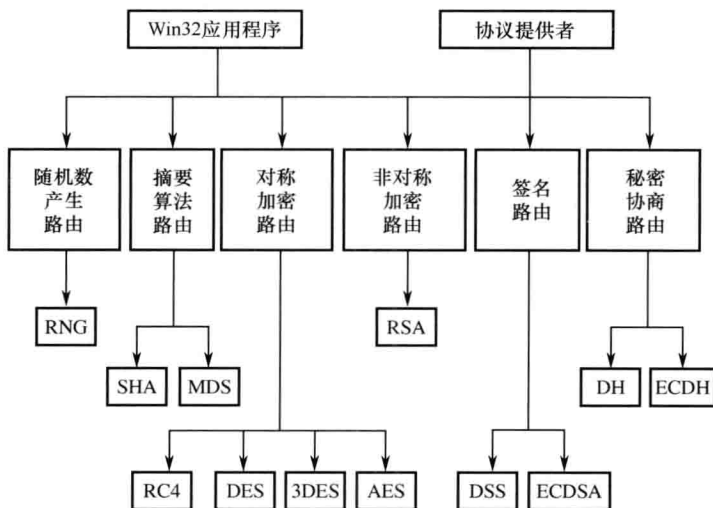


图 12-7 CNG 加密基元的结构和功能

3. CNG 密钥存储架构

CNG 提供的密钥存储模型能够满足现在和将来使用公钥或私钥进行加密的应用需求。密钥存储路由是这个模型的核心部件并在 Ncrypt.dll 实现。应用程序通过密钥存储路由访问安装在系统上的密钥存储提供者 (Key Storage Providers, KSP)，密钥存储路由封装了实现细节。图 12-8 显示了 CNG 密钥隔离架构的设计和功能。

依据 CC (Common Criteria) 通用标准的要求，长时间存在的密钥必须被隔离，使它们永远不会出现在应用程序中。目前，CNG 使用了 Windows Server 2008 和 Windows Vista 及后续 Windows 版本中默认安装的微软软件 KSP，以支持非对称私钥的存储。

默认情况下，在 Windows Server 2008 和 Windows Vista 及后续版本中启用了密钥隔离。此外，密钥隔离服务 (LSA 进程中) 不会加载第三方 KSP，只会加载微软的 KSP。

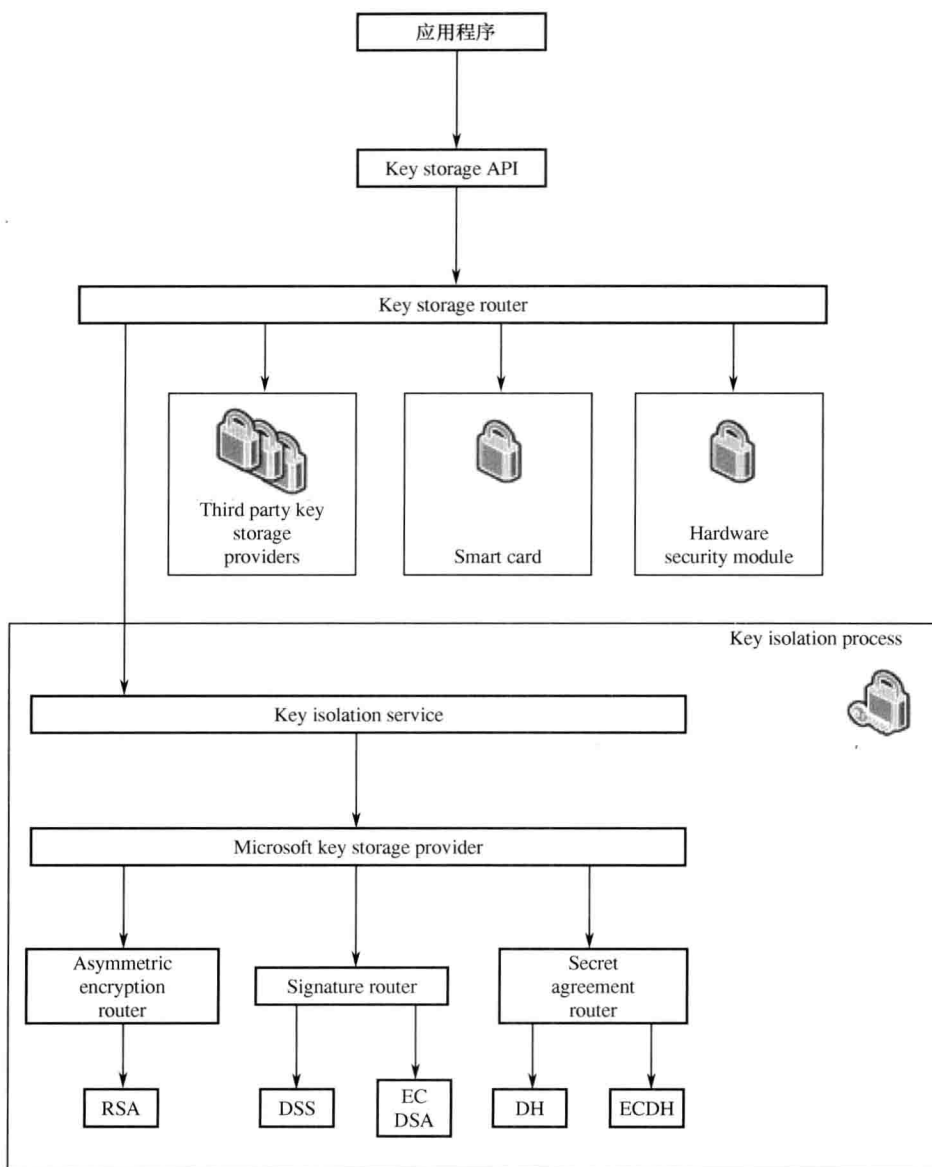


图 12-8 CNG 密钥隔离架构和功能

为最大限度地提高性能，使用 LSA 进程作为密钥隔离进程。私钥的访问都通过密钥存储路由，密钥存储路由输出了全面管理和使用私有密钥的函数。

CNG 存储私钥和公钥部分，密钥对的公共部分也由密钥隔离服务维护，并通过使用本地远程过程调用（LRPC）进行访问。密钥存储路由使用 LRPC 与密钥隔离进程交互。所有的私钥访问都要经过私钥路由，并由 CNG 审计。

CNG 支持以下类型的密钥：DH（Diffie-Hellman）、DSA（Digital Signature Algorithm）、RSA（PKCS#1）、部分 CryptoAPI 公私钥、ECC（Elliptic Curve Cryptography）。

与 CryptoAPI 相比，CNG 容器的不同之处如下。

① CNG 使用与 CSP（由 `Rsaenh.dll` 和 `Dssenh.dll` 创建）不同的密钥文件名，CSP 文件使用 `.key` 扩展名，CNG 密钥文件无此扩展名。

② CNG 完全支持 Unicode 密钥容器名，CNG 使用 Unicode 密钥容器名的哈希值，而 CryptoAPI 使用 ANSI 密钥容器名的哈希值。

③ CNG 对 RSA 密钥对的支持更灵活，如 CNG 支持超过 32 位长度的公钥 E 参数，并支持不同长度的 P 和 Q 值。

④ CryptoAPI 密钥容器文件保存在与用户 SID 相同名字的目录中，CNG 中不再如此，这样可以防止用户从一个域转移到另一个域时丢失密钥。

⑤ CNG 的 KSP 和密钥名字的长度限制为 `MAX_PATH` 个 Unicode 字符，CryptoAPI 的 CSP 和密钥名字的长度限制为 `MAX_PATH` 个 ANSI 字符。

⑥ CNG 支持用户定义的密钥属性，用户可以创建和关联密钥的用户属性，这些属性可以和持久密钥存放在一起。

当持久化密钥时，CNG 创建 2 个文件。第一个文件（总是创建）包含 CNG 新格式密钥，此文件不能被 CryptoAPI 的 CSP 使用；第二个文件以 CryptoAPI 密钥容器方式保存同样的密钥，第二个文件符合 `Rsaenh.dll` 使用规范。第二个文件只有在调用 `NCryptFinalizeKey` 时设置 `NCRYPT_WRITE_KEY_TO_LEGACY_STORE_FLAG` 参数，且密钥类型是 RSA 时才创建，当密钥为 DH 或 DSA 时，不创建第二个文件。

当应用程序访问存在的持久密钥时，CNG 首先打开 CNG 格式文件。如果此文件不存在，它会尝试打开 CryptoAPI 密钥容器中的匹配密钥。

4. 算法提供程序

如何使用 CNG 来执行各种常见的加密操作呢？首先需要有一个算法提供程序。BCrypt 定义的所有 CNG 对象均由 `BCRYPT_HANDLE` 标识，算法提供程序也不例外。`BCryptOpenAlgorithmProvider` 函数基于选择的算法和实现（可选）加载算法提供程序并返回一个句柄，以便在后续 CNG 函数调用中使用。不管是在用户模式下还是在内核模式下进行编码，BCrypt 还采用 Windows Driver Kit (WDK) 中的 `NTSTATUS` 类型指示错误信息。下面介绍如何将算法提供程序加载到内存中：

```
BCRYPT_HANDLE algorithmProvider = 0;
NTSTATUS status = ::BCryptOpenAlgorithmProvider(
    &algorithmProvider, algorithmName, implementation, flags);
if (NT_SUCCESS(status)) {
    // Use algorithm provider
}
```

在大多数情况下，将为 `implementation` 和 `flags` 参数传递 0。为 `implementation` 传递 0 表示应该为 `algorithmName` 参数标识的特定算法加载默认算法提供程序。`NT_SUCCESS` 宏用于指示状态值表示成功还是失败。

处理完算法提供程序后，必须通过将 `BCryptOpenAlgorithmProvider` 返回的句柄传递给 `BCryptCloseAlgorithmProvider` 函数将其卸载，如下所示：

```
status = ::BCryptCloseAlgorithmProvider(algorithmProvider, flags);
```



```
ASSERT(NT_SUCCESS(status));
```

当前没有为此函数定义 flags，因此必须为 flags 参数传递 0。

12.4.2 使用证书

在 CryptoAPI 中，可以使用 CertOpenSystemStore 函数打开证书库，然后使用 CertEnumCertificatesInStore 函数枚举证书，那么在 CNG 中，是否有类似的函数进行证书操作呢？

实际上，在 CryptoAPI 中使用的证书操作接口同样适用于 CNG 环境，也可以说，证书库操作函数不是 CryptoAPI 库的专用函数，无论 CSP 还是 CNG 环境，都可以使用证书操作函数对证书进行存取操作。证书和密钥通过一些属性关联起来，这些关联属性因 CSP 或 CNG 而有些差别。

在 Windows 7 环境下，一个证书可能关联一个 CryptoAPI 密钥，也可能关联一个 CNG 密钥，如何判断证书关联的密钥类型呢？可以使用命令行工具 certutil 进行信息展示。使用如下命令列出当前系统证书库中安装的证书：certutil -user -repairstore my*。

此命令显示用户证书及其关联的私钥信息。如果证书有关联的私钥，此命令会显示私钥提供者是 CSP 或 CNG。如果是 CNG 提供者，则会显示“提供程序 = XXX Key Storage Provider”；否则就是 CSP 提供者。如下显示了 CNG 提供者证书和密钥。

```
序列号: 0128
颁发者: CN=OpenSSL CA, S=Beijing, C=CN
NotBefore: 2009-11-24 14:04
NotAfter: 2019-11-12 14:04
使用者: CN=ldapclient, S=Beijing, C=CN
非根证书
模板:
证书哈希(sha1): cf 65 1a ad 03 c4 a9 41 47 3a 7e 2a a1 19 3e de a5 00 a9 c8
密钥容器 = {40660087-6C63-4054-BD3C-982EAA77244D}
唯一容器名称:
    11cdae81ff4a19f0bc6e6c783f2be964_c40d85d9-e84f-4c76-b11b-df34193b54ed
提供程序 = Microsoft Software Key Storage Provider
通过了加密测试
```

如果要查看系统安装的所有 CSP 和 CNG 提供者，可以使用命令“certutil -csplist”，如果“提供程序类型”信息为空，表示为 CNG 提供者，如下所示。

```
提供程序名称: Microsoft Enhanced Cryptographic Provider v1.0
提供程序类型: 1 - PROV_RSA_FULL
提供程序名称: Microsoft Enhanced RSA and AES Cryptographic Provider
提供程序类型: 24 - PROV_RSA_AES
提供程序名称: Microsoft Strong Cryptographic Provider
提供程序类型: 1 - PROV_RSA_FULL
提供程序名称: Microsoft Software Key Storage Provider
提供程序名称: Microsoft Smart Card Key Storage Provider
```

前 3 个是 CSP 提供者, 后 2 个是 CNG 提供者。注意, CNG 提供者名称中有“Key Storage Provider”。

在 Windows 7 环境下, 导入 PKCS#12 (.pfx 后缀) 的证书私钥到微软的 CNG 密钥存储“Microsoft Software Key Storage Provider”提供者的命令行:

```
certutil -csp "Microsoft Software Key Storage Provider" -p 123456 -user -f -importPFX mycertkey.pfx
```

其中, “-csp”指定密钥提供者, 此处使用微软 CNG 密钥存储容器, “-p”指定 mycertkey.pfx 文件的保护口令, “-user”为当前用户存储区, “-f”表示强制覆盖, 如果已经存在对应证书私钥, 则强制覆盖, “-importPFX”指定要导入的文件。

由于证书既可能关联 CSP 私钥, 也可能关联 CNG 私钥, 所以在程序中需要区分, 以使用不同的密钥操作函数。为了获取证书关联的 CNG 私钥, 同样需要使用 CryptAcquireCertificatePrivateKey 函数。在 Windows Vista 以后的版本中, 微软扩展了此接口, 使其可以支持获取 CNG 密钥。其函数原型如下:

```
BOOL WINAPI CryptAcquireCertificatePrivateKey(
    __in    PCCERT_CONTEXT pCert,
    __in    DWORD dwFlags,
    __in    void *pvReserved,
    __out   HCRYPTPROV_OR_NCRYPT_KEY_HANDLE *phCryptProvOrNCryptKey,
    __out   DWORD *pdwKeySpec,
    __out   BOOL *pfCallerFreeProvOrNCryptKey);
```

在 dwFlags 中, 可以指定如下标志, 使 CryptAcquireCertificatePrivateKey 函数尝试 CNG 密钥:

① CRYPT_ACQUIRE_ALLOW_NCRYPT_KEY_FLAG: 首先使用 CryptoAPI 方式获取私钥, 如果失败, 则使用 CNG 方式获取私钥。如果使用 CNG 方式获得私钥, 则 pdwKeySpec 值会被设置为 CERT_NCRYPT_KEY_SPEC。

② CRYPT_ACQUIRE_ONLY_NCRYPT_KEY_FLAG: 只使用 CNG 方式获取私钥, pdwKeySpec 中会被设置为 CERT_NCRYPT_KEY_SPEC。

由于证书使用方式与 CryptoAPI 相同, 请参考 12.1 节的内容。

12.4.3 使用私钥

与 CSP 使用密钥方式类似, 在使用密钥前, 需要先获得密钥句柄, 通过密钥句柄完成签名、加密等密钥运算。

在 12.1 节演示了如何获得 PCCERT_CONTEXT 类型的证书指针, 下面介绍通过证书指针获得 CNG 密钥句柄, 然后进行签名过程。

12.4.3.1 私钥签名过程

① 通过 CryptAcquireCertificatePrivateKey 获得私钥句柄。

② 如果关联私钥为 CNG 私钥, 则调用 SignWithCngKey 函数进行签名。

③ 在 `SignWithCngKey` 函数中:

- a. 首先使用 `BCryptOpenAlgorithmProvider` 获得哈希算法提供者, 使用了默认提供者。
- b. 通过 `BCryptGetProperty` 函数获得哈希对象内存大小。
- c. 申请哈希对象内存空间。
- d. 通过 `BCryptGetProperty` 函数获得哈希算法结果长度。
- e. 申请哈希结果内存空间。
- f. 通过 `BCryptCreateHash` 函数创建哈希对象。
- g. 通过 `BCryptHashData` 函数计算数据的哈希值。
- h. 通过 `BCryptFinishHash` 获得哈希结果。
- i. 调用 `NCryptSignHash` 计算签名结果数据长度。
- j. 申请签名结果空间。
- k. 调用 `NCryptSignHash` 计算签名结果数据。
- l. 清空申请的内存空间。

至此, 在 `pbSignature` 中保存了签名结果, 签名结果数据的长度为 `cbSignature`。

12.4.3.2 私钥签名示例程序

```
PCCERT_CONTEXT pDesiredCert = ...; // 已经通过证书库获得
HCRYPTPROV_OR_NCRYPT_KEY_HANDLE hcngKey;
DWORD dwSpec = 0;
BOOL fCallerFreeProvOrNCryptKey;
BOOL rc = CryptAcquireCertificatePrivateKey(pDesiredCert,
    CRYPT_ACQUIRE_USE_PROV_INFO_FLAG |
    CRYPT_ACQUIRE_ALLOW_NCRYPT_KEY_FLAG,
    NULL, &hcngKey, &dwSpec, &fCallerFreeProvOrNCryptKey);
if (!rc) {
    // CryptAcquireCertificatePrivateKey 打开私钥错误, 可能证书没用关联私钥
    return;
}
if (dwSpec == CERT_NCRYPT_KEY_SPEC) {
    // 说明证书关联的是 CNG 类型私钥, 调用 SignWithCngKey 函数进行签名
    SignWithCngKey((NCRYPT_KEY_HANDLE)hcngKey);
    printf("Cert key is CNG key. \n");
}

// 函数 SignWithCngKey 实现如下
#define KNOWN_DATA "SignThis"
#define KNOWN_DATALEN 8
static void SignWithCngKey(NCRYPT_KEY_HANDLE hKey)
{
    BCrypt_ALG_HANDLE hAlgorithm = NULL;
```

```

BCRYPT_HASH_HANDLE      hHash          = NULL;
NTSTATUS                  Status          = STATUS_UNSUCCESSFUL;
SECURITY_STATUS          secStatus       = ERROR_SUCCESS;
DWORD                   cbHash          = 0,
                        cbData          = 0,
                        cbHashObject     = 0;
PBYTE                   pbHashObject     = NULL;
PBYTE                   pbHash          = NULL,
                        pbBlob          = NULL;
PBYTE                   pbSignature      = NULL;
DWORD                   cbSignature      = 0,
                        cbBlob          = 0;

BCRYPT_PKCS1_PADDING_INFO PKCS1PaddingInfo= {0};
// 打开算法提供者句柄
if(!NT_SUCCESS(Status = BCryptOpenAlgorithmProvider(
    &hAlgorithm,  NCRYPT_SHA1_ALGORITHM,
    NULL, 0)))
{
    wprintf(L"Error 0x%x returned by BCryptOpenAlgorithmProvider\n", Status);
    goto Cleanup;
}
if(!NT_SUCCESS(Status = BCryptGetProperty(
    hAlgorithm,  BCRYPT_OBJECT_LENGTH,
    (PBYTE)&cbHashObject,  sizeof(DWORD),  &cbData,  0)))
{
    wprintf(L"***** Error 0x%x returned by BCryptGetProperty\n", Status);
    goto Cleanup;
}
pbHashObject = (PBYTE)HeapAlloc (GetProcessHeap (), 0,cbHashObject);
if(NULL == pbHashObject)
{
    wprintf(L"***** memory allocation failed\n");
    goto Cleanup;
}
//获取 HASH 缓冲区空间大小
if(!NT_SUCCESS(Status = BCryptGetProperty(
    hAlgorithm,  BCRYPT_HASH_LENGTH,
    (PBYTE)&cbHash,  sizeof(DWORD),  &cbData,  0)))
{
    wprintf(L"***** Error 0x%x returned by BCryptGetProperty\n", Status);
    goto Cleanup;
}
pbHash = (PBYTE)HeapAlloc (GetProcessHeap (), 0, cbHash);

```

```
if(NULL == pbHash)
{
    wprintf(L"**** memory allocation failed\n");
    goto Cleanup;
}
if(!NT_SUCCESS(Status = BCryptCreateHash(
    hAlgorithm, &hHash, pbHashObject,
    cbHashObject, NULL, 0, 0)))
{
    wprintf(L"**** Error 0x%x returned by BCryptCreateHash\n", Status);
    goto Cleanup;
}
if(!NT_SUCCESS(Status = BCryptHashData(
    hHash, (PBYTE)KNOWN_DATA,
    KNOWN_DATALEN, 0)))
{
    wprintf(L"**** Error 0x%x returned by BCryptHashData\n", Status);
    goto Cleanup;
}
if(!NT_SUCCESS(Status = BCryptFinishHash(
    hHash, pbHash, cbHash, 0)))
{
    wprintf(L"**** Error 0x%x returned by BCryptFinishHash\n", Status);
    goto Cleanup;
}
PKCS1PaddingInfo.pszAlgId = NCrypt_SHA1_ALGORITHM;
if(FAILED(secStatus = NCryptSignHash(
    hKey, &PKCS1PaddingInfo, pbHash, cbHash,
    NULL, 0, &cbSignature, NCrypt_PAD_PKCS1_FLAG)))
{
    wprintf(L"**** Error 0x%x returned by NCryptSignHash\n", secStatus);
    goto Cleanup;
}
pbSignature = (PBYTE)HeapAlloc (GetProcessHeap (), 0, cbSignature);
if(NULL == pbSignature)
{
    wprintf(L"**** memory allocation failed\n");
    goto Cleanup;
}
if(FAILED(secStatus = NCryptSignHash(
    hKey, &PKCS1PaddingInfo, pbHash, cbHash,
    pbSignature, cbSignature, &cbSignature,
    NCrypt_PAD_PKCS1_FLAG)))
```

```

{
    wprintf(L"**** Error 0x%x returned by NCryptSignHash\n", secStatus);
    goto Cleanup;
}

Cleanup:
    if(hHash) { BCryptDestroyHash(hHash); }
    if(hAlgorithm) { BCryptCloseAlgorithmProvider(hAlgorithm,0); }
    if(pbHashObject) { HeapFree(GetProcessHeap(), 0, pbHashObject); }
    if(pbHash) { HeapFree(GetProcessHeap(), 0, pbHash); }
    if(pbSignature) { HeapFree(GetProcessHeap(), 0, pbSignature); }
    if(pbBlob) { HeapFree(GetProcessHeap(), 0, pbBlob); }
    return;
}

```

12.5 PC/SC

12.5.1 PC/SC 简介

PC/SC 即个人计算机 (Personal Computer) / 智能卡 (Smart Card)，由微软公司与世界其他著名的智能卡厂商组成的 PC/SC 工作组提出，它是为智能卡访问 Windows 平台定义的一种标准结构。

PC/SC 标准是一个基于 Windows 平台的标准用户接口 (API)，提供了一个从个人计算机 (Personal Computer) 到智能卡 (Smart Card) 的整合环境，为集成电路卡 (ICC) 与个人计算机系统设计的交互规范，让智能卡进入 PC 世界变得容易。PC/SC 的主要优点就是让应用程序不必为了与智能卡通信而去了解智能卡读卡器的细节，而且应用程序还能适用于任何遵从 PC/SC 标准的读卡器。

Windows 系统中，PC/SC 功能通过 Windows 智能卡 (WinSCard) 客户端 API 提供给应用程序，该 API 在 winscard.dll 和 scarddlg.dll 中实现。

由于智能卡是一种共享资源，在任何给定时间都可能多个应用程序试图与该卡进行通信，与该卡的通信必须是串行的，因此要使用资源管理器模型。资源管理器强制采用简化的、类似于数据库的锁定模型。在 PC/SC 中，持有给定智能卡读卡器（也称为接口设备或 IFD）的锁，就等于掌握了智能卡资源。

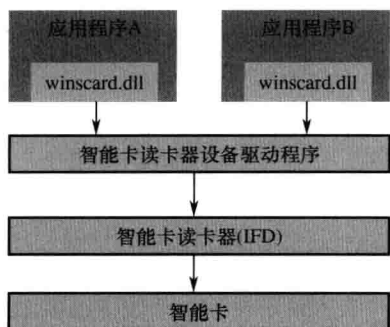


图 12-9 智能卡调用分层结构

图 12-9 说明了 PC/SC 组件堆栈。PC/SC 客户端代码是在应用程序进程中加载的。资源管理器是一种系统服务，是独立于客户端的。读卡器设备驱动程序是唯一在内核模式下运行的 PC/SC 堆栈组件。读卡器驱动程序被资源管理器独占加载，以防止应用程序忽略事务模型。

智能卡在 Windows 中的典型用途通常与身份验证相关。例如，它们可以代替密码进行登录。这些智能卡身份

验证方案是基于加密技术的，这就导致一直以来，在 Windows 中添加对新型智能卡的支持始终要求供应商部署一个 CSP（加密服务提供商）或 CNG（下一代加密接口）的插件，该插件实现了 CryptoAPI 1.0 或 CAPI 2.0 的接口。

为了简化 CSP/CNG 插件的编写，微软提供了智能卡的微型驱动（mini-driver）接口，通过编写 mini-driver，应用程序可以调用基于微软标准的 Microsoft Smart Card Base CSP 和 KSP 服务使用智能卡。图 12-10 说明了智能卡的 mini-driver 与基于 CSP 的应用程序的组成结构。应用程序通过 CAPI 接口，调用 Smart Card Base Cryptographic 服务提供者，智能卡 CSP 提供者使用智能卡 mini-driver 与智能卡设备交互，完成密码运算，而 mini-driver 使用 WinScard 接口与智能卡交互。从图中可以看出，通过智能卡 mini-driver 驱动，应用程序可以完全利用 CSP/CNG 提供的接口完成与智能卡的交互。

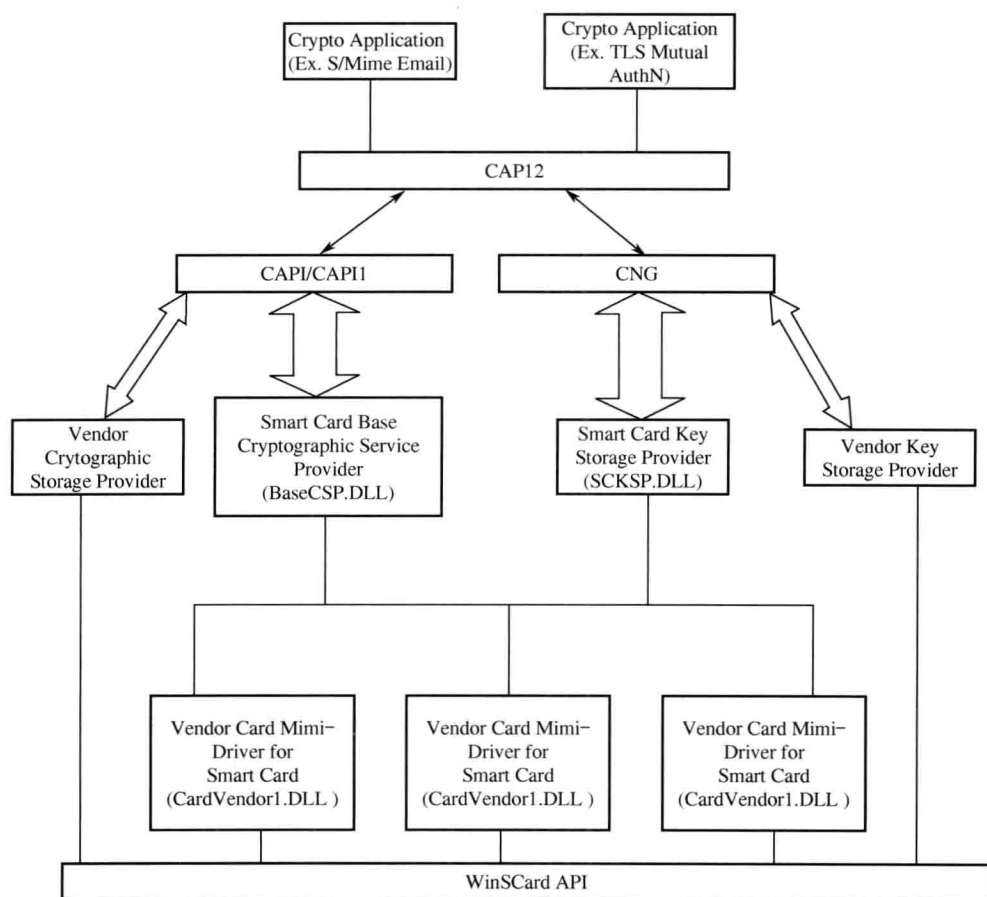


图 12-10 智能卡应用与 mini-driver 的交互结构

如果不使用 CSP/CNG 提供的接口，应用程序调用智能卡功能必须通过发送指令形式，使用 SCardTransmit 函数发送指令，具体示例代码如下：

```

LONG rv;
DWORD dwRecvLength;

```

```

BYTE pbRecvBuffer[258];
BYTE cmd2[] = { 0x00, 0x00, 0x00, 0x00 };
dwRecvLength = sizeof(pbRecvBuffer);
rv = SCardTransmit(hCard, &pioSendPci, cmd2, sizeof(cmd2), NULL,
                  pbRecvBuffer, &dwRecvLength);

```

12.5.2 使用证书

为了便于说明 winscard 接口和 CryptoAPI 接口的联合调用，Windows Platform SDK 中自带一个代码示例，把智能卡证书导入到证书库。

在该示例中，首先通过 SCardEstablishContext 获得资源管理控制器上下文，接着调用 SCardListReaders 列出所有的已安装的读卡器，然后调用自定义函数 SCardPropCert 完成证书导入。SCardPropCert 返回后，调用 SCardFreeMemory 释放申请的内存，调用 SCardReleaseContext 释放资源管理控制器上下文。

在 SCardPropCert 函数中，执行如下操作：

- ① 获取读卡器的数量。
- ② 分配存储读卡器状态数据数组，用于获取读卡器信息。
- ③ 调用函数 SCardGetStatusChange 获取读卡器状态。
- ④ 通过 SCARD_READERSTATE 的 dwEventState 字段，判断 SCARD_STATE_PRESENT 位是否被置位，若置位表示存在智能卡读卡器。

⑤ 对存在的读卡器，调用 SCardListCards 函数获取智能卡名称。

⑥ 调用函数 SCardGetCardTypeProviderName 获取智能卡 CSP 提供者名称。

⑦ 分配容器名称存储空间内存，然后构建容器名称。

⑧ 调用函数 CryptAcquireContext 打开 CSP 提供者上下文句柄。

⑨ 调用自定义函数 CryptPropCert 完成证书导出。

⑩ 调用 CryptReleaseContext 释放 CSP 提供者上下文句柄。

⑪ 释放申请的内存空间。

在 CryptPropCert 函数中，执行如下操作：

① 调用函数 CryptGetProvParam 获取密钥容器名称。

② 调用自定义函数 GetCert 分别获得签名私钥和加密私钥对应证书。

③ 调用自定义函数 AddCert 把证书加入个人证书库中。

④ 释放申请的内存空间。

在 GetCert 函数中，执行如下操作：

① 调用函数 CryptGetUserKey 获得容器中的密钥句柄。

② 调用函数 CryptGetKeyParam 获取证书数据长度。

③ 为证书数据分配内存空间。

④ 调用函数 CryptGetKeyParam 获取证书数据。

⑤ 释放申请内存空间。

⑥ 调用函数 CryptDestroyKey 释放获得证书句柄。

在 AddCert 函数中，执行如下操作：

- ① 调用函数 `CertOpenStore` 打开证书库句柄。
 - ② 以获取的证书数据为参数，调用函数 `CertCreateCertificateContext` 生成证书对象。
 - ③ 调用函数 `CertSetCertificateContextProperty` 设置证书的易用名、提供者名称、密钥类型、容器名。
 - ④ 调用函数 `CertAddCertificateContextToStore` 把证书加入到证书库中。
 - ⑤ 释放证书对象。
 - ⑥ 关闭证书库句柄。
- 该示例代码如下：

```
#define _WIN32_WINNT 0x0400
#include <tchar.h>
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>
#include <wincard.h>
#include <wincrypt.h>
// Macros
#define MALLOC(size) ((LPBYTE) LocalAlloc(LPTR, size))
#define FREE(buffer) (LocalFree((LPBYTE) buffer))
// Prototypes
LONG SCardPropCert (IN SCARDCONTEXT hContext,
    IN LPCTSTR mszReaderNames, IN LPCTSTR szStoreName);
LONG CryptPropCert (IN HCRYPTPROV hCryptProv,
    IN LPCTSTR szCSPName, IN LPCTSTR szStoreName);
LONG GetCert (IN HCRYPTPROV hCryptProv,
    IN DWORD dwKeySpec, OUT LPBYTE * lpIpbCert,
    OUT DWORD * lpdwCertLength);
LONG AddCert (IN HCRYPTPROV hCryptProv,
    IN LPBYTE lpbCert, IN DWORD dwCertLength,
    IN DWORD dwKeySpec, IN LPCWSTR szCertFriendlyName,
    IN LPCWSTR zContainerName, IN LPCWSTR szCSPName,
    IN LPCWSTR szStoreName);
int __cdecl _tmain (int argc, _TCHAR * argv[])
{
    LONG lResult;
    DWORD dwNumReaders = 0;
    SCARDCONTEXT hContext = NULL;
    LPTSTR mszReaderNames = NULL;
    __try {
        // Establish context with the resource manager.
        lResult = SCardEstablishContext(SCARD_SCOPE_USER,
            NULL, NULL, &hContext);
        if (lResult != SCARD_S_SUCCESS) { __leave; }
```

```

    // Get the list of reader(s) associated with the specified group(s).
    // Note: The buffer is automatically allocated and must be freed
    //       by SCardFreeMemory().
    DWORD dwAutoAllocate = SCARD_AUTOALLOCATE;
    HRESULT = SCardListReaders(hContext, SCARD_DEFAULT_READERS,
        (LPTSTR) &mszReaderNames, &dwAutoAllocate);
    if (HRESULT != SCARD_S_SUCCESS) { __leave; }
    // Propagate all digital certificate(s) found in all reader(s) to the
    // local "My" store.
    HRESULT = SCardPropCert(hContext, mszReaderNames, _T("My"));
}
__finally {
    LONG lReturn;
    // Don't forget to free resources, if allocated.
    if (mszReaderNames != NULL) {
        lReturn = SCardFreeMemory(hContext, (LPVOID) mszReaderNames);
        // If successful so far, then capture the return code
        // from SCFree(); otherwise, don't bother.
        if (HRESULT == SCARD_S_SUCCESS) { lReturn = lReturn; }
    }
    if (hContext != NULL) {
        lReturn = SCardReleaseContext(hContext);
        // If successful so far, then capture the SCardReleaseContext()
        // return code; otherwise, don't bother.
        if (HRESULT == SCARD_S_SUCCESS) { lReturn = lReturn; }
    }
}
// Inform user if an error had occurred.
if (HRESULT != SCARD_S_SUCCESS) {
    _tprintf(_T("\nError [0x%x]: Program terminated abnormally.\n"), HRESULT);
}
return lReturn;
}

LONG SCardPropCert (IN SCARDCONTEXT hContext,
    IN LPCTSTR mszReaderNames, IN LPCTSTR szStoreName)
{
    LONG lResult;
    LPSCARD_READERSTATE lpReaderStates = NULL;
    // Make sure pointer parameters are not NULL.
    if (mszReaderNames == NULL || szStoreName == NULL) {
        return SCARD_E_INVALID_PARAMETER;
    }
    __try {

```

```

DWORD dwNumReaders;
LPCTSTR szReaderName;
// Count number of readers.
for (dwNumReaders = 0, szReaderName = mszReaderNames;
     *szReaderName != _T('\0'); dwNumReaders++) {
    szReaderName += lstrlen(szReaderName) + 1;
}
// Allocate memory for SCARD_READERSTATE array.
lpReaderStates = (LPSCARD_READERSTATE)
    MALLOC(dwNumReaders * sizeof(SCARD_READERSTATE));
if (lpReaderStates == NULL) {
    lResult = SCARD_E_NO_MEMORY;
    __leave;
}
// Prepare state array.
ZeroMemory((LPVOID) lpReaderStates,
            dwNumReaders * sizeof(SCARD_READERSTATE));
DWORD i;
for (i = 0, szReaderName = mszReaderNames;
     i < dwNumReaders; i++) {
    lpReaderStates[i].szReader = (LPCTSTR) szReaderName;
    lpReaderStates[i].dwCurrentState = SCARD_STATE_UNAWARE;
    szReaderName += lstrlen(szReaderName) + 1;
}
// Initialize card status.
lResult = SCardGetStatusChange(hContext, INFINITE,
                               lpReaderStates, dwNumReaders);
if (lResult != SCARD_S_SUCCESS) { __leave; }
// For each card found, find the proper CSP and propagate the
// certificate(s) to the specified local store.
for (i = 0; i < dwNumReaders && lResult == SCARD_S_SUCCESS; i++) {
    DWORD dwAutoAllocate;
    LPTSTR szCardName = NULL;
    LPTSTR szCSPName = NULL;
    LPTSTR szContainerName = NULL;
    HCRYPTPROV hCryptProv = NULL;
    __try {
        // Card in this reader?
        if (!(lpReaderStates[i].dwEventState & SCARD_STATE_PRESENT)){
            // No card in this reader.
            continue;
        }
        // Get card name.
    }
}

```

```

dwAutoAllocate = SCARD_AUTOALLOCATE;
IResult = SCardListCards(hContext, lpReaderStates[i].rgbAtr,
    NULL, 0, (LPTSTR) &szCardName, &dwAutoAllocate);
if (IResult != SCARD_S_SUCCESS) { __leave; }
// Get card's CSP name.
dwAutoAllocate = SCARD_AUTOALLOCATE;
IResult = SCardGetCardTypeProviderName(hContext, szCardName,
    SCARD_PROVIDER_CSP, (LPTSTR) &szCSPName,
    &dwAutoAllocate);
if (IResult != SCARD_S_SUCCESS) { __leave; }
// Prepare fully qualified container name.
szContainerName = (LPTSTR) MALLOC((sizeof(_T("\\\\.\\")) +
    strlen(lpReaderStates[i].szReader) +
    sizeof(_T("\\0")) * sizeof(TCHAR)));
if (szContainerName == NULL) {
    IResult = SCARD_E_NO_MEMORY;
    __leave;
}
wsprintf(szContainerName, _T("\\\\.\\"), lpReaderStates[i].szReader);
// Obtain the crypto context.
// CRYPT_SILENT forces the CSP to raise no UI. The fully qualified
// container name indicates which reader to connect to, so the
// user should not be prompted to insert or select a card.
if (!CryptAcquireContext(&hCryptProv, szContainerName,
    szCSPName, PROV_RSA_FULL, CRYPT_SILENT)) {
    IResult = GetLastError();
    __leave;
}
// Propagate the cert.
IResult = CryptPropCert(hCryptProv, szCSPName, szStoreName);
}
__finally {
    LONG IReturn;
    // Don't forget to free resources, if allocated.
    if (hCryptProv != NULL) {
        if (!CryptReleaseContext(hCryptProv, 0)) {
            if (IResult == SCARD_S_SUCCESS) {
                IResult = GetLastError();
            }
        }
    }
}
if (szContainerName != NULL) {
    FREE((LPVOID) szContainerName);
}

```

```

    }
    if (szCSPName != NULL) {
        IReturn = SCardFreeMemory(hContext, (LPVOID) szCSPName);
        if (IResult == SCARD_S_SUCCESS) {
            IResult = IReturn;
        }
    }
    if (szCardName != NULL)
    {
        IReturn = SCardFreeMemory(hContext, (LPVOID) szCardName);
        if (IResult == SCARD_S_SUCCESS) {
            IResult = IReturn;
        }
    }
}

}

}

__finally {
    // Don't forget to free resources, if allocated.
    if (lpReaderStates != NULL) {
        FREE((LPVOID) lpReaderStates);
    }
}

return IResult;
}

LONG CryptPropCert (IN HCRYPTPROV hCryptProv,
    IN LPCTSTR szCSPName, IN LPCTSTR szStoreName)
{
    LONG IResult;
    LPCTSTR szContainerName = NULL;
    // Make sure pointer parameters are not NULL.
    if (szCSPName == NULL || szStoreName == NULL) {
        return SCARD_E_INVALID_PARAMETER;
    }
    __try {
        // Query length of key container name.
        DWORD cbContainerName = 0;
        if (!CryptGetProvParam(hCryptProv, PP_CONTAINER,
            NULL, // NULL to query container name length
            &cbContainerName, 0)) {
            IResult = GetLastError();
            __leave;
        }
    }
}

```

```

// Allocate memory for key container name.
szContainerName = (LPCTSTR) MALLOC(cbContainerName);
// Now get the key container name.
if (!CryptGetProvParam(hCryptProv, PP_CONTAINER,
    (PBYTE) szContainerName, &cbContainerName, 0)) {
    HRESULT = GetLastError();
    __leave;
}
// For each key pair found in the smart card, store the corresponding
// digital certificate to the specified local store.
const DWORD rgdwKeys[] = {AT_KEYEXCHANGE, AT_SIGNATURE};
const DWORD cdwKeys = sizeof(rgdwKeys) / sizeof(rgdwKeys[0]);
for (DWORD i = 0; i < cdwKeys; i++) {
    DWORD dwCertLength = 0;
    LPBYTE lpbCert = NULL;
    LPWSTR wszCertFriendlyName = NULL;
    LPWSTR wszContainerName = NULL;
    LPWSTR wszCSPName = NULL;
    LPWSTR wszStoreName = NULL;
    __try {
        // Get the certificate data.
        HRESULT = GetCert(hCryptProv, rgdwKeys[i],
            &lpbCert, &dwCertLength);
        if (HRESULT != SCARD_S_SUCCESS) {
            if (HRESULT == NTE_NO_KEY) {
                // We are OK if there is no key of such type.
                // It means there is nothing to do.
                HRESULT = SCARD_S_SUCCESS;
            }
            __leave;
        }
        // Allocate memory for UNICODE strings.
        TCHAR szCertFriendlyName[] = "";
        DWORD cchCertFriendlyName = (lstrlen(szCertFriendlyName) + 1)
            * sizeof(WCHAR);
        DWORD cchContainerName = (lstrlen(szContainerName) + 1)
            * sizeof(WCHAR);
        DWORD cchCSPName = (lstrlen(szCSPName) + 1) * sizeof(WCHAR);
        DWORD cchStoreName = (lstrlen(szStoreName) + 1) * sizeof(WCHAR);
        wszCertFriendlyName = (LPWSTR) MALLOC(cchCertFriendlyName);
        wszContainerName = (LPWSTR) MALLOC(cchContainerName);
        wszCSPName = (LPWSTR) MALLOC(cchCSPName);
        wszStoreName = (LPWSTR) MALLOC(cchStoreName);
    }
}

```

```

        if (wszCertFriendlyName == NULL || wszContainerName == NULL ||
            wszCSPName == NULL || wszStoreName == NULL) {
            HRESULT = SCARD_E_NO_MEMORY;
            __leave;
        }
        // Setup UNICODE strings.
#ifdef _UNICODE
        lstrcpy(wszCertFriendlyName, szCertFriendlyName);
        lstrcpy(wszContainerName, szContainerName);
        lstrcpy(wszCSPName, szCSPName);
        lstrcpy(wszStoreName, szStoreName);
#else
        mbstowcs(wszCertFriendlyName, szCertFriendlyName,
            cchCertFriendlyName);
        mbstowcs(wszContainerName, szContainerName, cchContainerName);
        mbstowcs(wszCSPName, szCSPName, cchCSPName);
        mbstowcs(wszStoreName, szStoreName, cchStoreName);
#endif

        // Add the certificate to the specified local store.
        HRESULT = AddCert(hCryptProv, lpbCert,
            dwCertLength, rgdwKeys[i], wszCertFriendlyName,
            wszContainerName, wszCSPName, wszStoreName);
        if (HRESULT != SCARD_S_SUCCESS) {
            __leave;
        } else {
            _tprintf(
                _T("\nPropagated cert from %s to \"%s\" certificate store.\n"),
                szCSPName, szStoreName);
        }
    }
    __finally {
        // Don't forget to free resources, if allocated.
        if (lpbCert != NULL) { FREE(lpbCert); }
        if (wszCertFriendlyName != NULL) { FREE(wszCertFriendlyName); }
        if (wszContainerName != NULL) { FREE(wszContainerName); }
        if (wszCSPName != NULL) { FREE(wszCSPName); }
        if (wszStoreName != NULL) { FREE(wszStoreName); }
        if (HRESULT != SCARD_S_SUCCESS) { __leave; }
    }
}

__finally {
    // Don't forget to free resources, if allocated.

```

```

        if (szContainerName != NULL) { FREE(szContainerName); }
    }
    return HRESULT;
}

LONG GetCert (IN HCRYPTPROV hCryptProv, IN DWORD dwKeySpec,
    OUT LPBYTE * lpPbCert, OUT DWORD * lpdwCertLength)
{
    LONG HRESULT = SCARD_S_SUCCESS;
    HCRYPTKEY hCryptKey = NULL;
    LPBYTE lpbCert = NULL;
    DWORD dwCertLength = 0;
    // Make sure pointer parameters are not NULL.
    if (lpPbCert == NULL || lpdwCertLength == NULL) {
        return SCARD_E_INVALID_PARAMETER;
    }
    __try {
        // Get key handle.
        if (!CryptGetUserKey(hCryptProv, dwKeySpec, &hCryptKey)) {
            HRESULT = GetLastError();
            __leave;
        }
        // Query certificate data length.
        if (!CryptGetKeyParam(hCryptKey, KP_CERTIFICATE,
            NULL, // NULL to query certificate data length
            &dwCertLength, 0)) {
            // We expect ERROR_MORE_DATA. If that's not the case, then
            // something is not right.
            HRESULT = GetLastError();
            if (HRESULT == ERROR_MORE_DATA) {
                HRESULT = SCARD_S_SUCCESS;
            } else {
                __leave;
            }
        }
        // Allocate memory for certificate data.
        lpbCert = (LPBYTE) MALLOC(dwCertLength);
        if (lpbCert == NULL) {
            HRESULT = SCARD_E_NO_MEMORY;
            __leave;
        }
        // Now read the certificate data.
        if (!CryptGetKeyParam(hCryptKey, KP_CERTIFICATE,
            lpbCert, &dwCertLength, 0)) {

```



```

        HRESULT = GetLastError();
        __leave;
    }
}
__finally {
    // Don't forget to free resources, if allocated.
    if (HRESULT == SCARD_S_SUCCESS) {
        *lpbCert = lpbCert;
        *lpdwCertLength = dwCertLength;
    } else if (lpbCert != NULL) {
        FREE(lpbCert);
    }
    if (hCryptKey != NULL) {
        if (!CryptDestroyKey(hCryptKey)) {
            if (HRESULT == SCARD_S_SUCCESS) {
                HRESULT = GetLastError();
            }
        }
    }
}
return HRESULT;
}

LONG AddCert (IN HCRYPTPROV hCryptProv,
              IN LPBYTE lpbCert,
              IN DWORD dwCertLength,
              IN DWORD dwKeySpec,
              IN LPCWSTR wszCertFriendlyName,
              IN LPCWSTR wszContainerName,
              IN LPCWSTR wszCSPName,
              IN LPCWSTR wszStoreName)
{
    LONG HRESULT = SCARD_S_SUCCESS;
    HCERTSTORE hCertStore = NULL;
    PCCERT_CONTEXT pCertContext = NULL;
    // Make sure pointer parameters are not NULL.
    if (lpbCert == NULL || wszContainerName == NULL ||
        wszCSPName == NULL || wszStoreName == NULL) {
        return SCARD_E_INVALID_PARAMETER;
    }
    __try {
        // Open the user's specified store for writing.
        hCertStore = CertOpenStore(CERT_STORE_PROV_SYSTEM_W,

```

```

    0, hCryptProv,
    CERT_STORE_NO_CRYPT_RELEASE_FLAG |
    CERT_SYSTEM_STORE_CURRENT_USER,
    wszStoreName);
if (NULL == hCertStore) {
    HRESULT = GetLastError();
    __leave;
}
// Build certificate context for this certificate.
pCertContext = CertCreateCertificateContext(X509_ASN_ENCODING,
    lpbCert, dwCertLength);
if (pCertContext == NULL) {
    HRESULT = GetLastError();
    __leave;
}
// Add the friendly name, if provided.
if (wszCertFriendlyName != NULL) {
    CRYPT_DATA_BLOB DataBlob;
    ZeroMemory((PVOID)&DataBlob, sizeof(CRYPT_DATA_BLOB));
    DataBlob.cbData = (wcslen(wszCertFriendlyName) + 1) * sizeof(WCHAR);
    DataBlob.pbData = (LPBYTE) wszCertFriendlyName;
    if (!CertSetCertificateContextProperty(pCertContext,
        CERT_FRIENDLY_NAME_PROP_ID, 0,
        (const void *) &DataBlob)) {
        HRESULT = GetLastError();
        __leave;
    }
}
// Add the CSP & key container info. This is used by CAPI to load the
// CSP and find the keyset when the user indicates this certificate.
CRYPT_KEY_PROV_INFO KeyProvInfo;
ZeroMemory((PVOID)&KeyProvInfo, sizeof(CRYPT_KEY_PROV_INFO));
KeyProvInfo.pwszContainerName = (LPWSTR) wszContainerName;
KeyProvInfo.pwszProvName = (LPWSTR) wszCSPName;
KeyProvInfo.dwProvType = PROV_RSA_FULL;
KeyProvInfo.dwFlags = 0;
KeyProvInfo.dwKeySpec = dwKeySpec;
if (!CertSetCertificateContextProperty(pCertContext,
    CERT_KEY_PROV_INFO_PROP_ID,
    0, (const void *) &KeyProvInfo)) {
    HRESULT = GetLastError();
    __leave;
}

```

```

    // Put the cert in the store!
    if (!CertAddCertificateContextToStore(hCertStore, pCertContext,
        CERT_STORE_ADD_REPLACE_EXISTING, //or CERT_STORE_ADD_NEW
        NULL)) {
        HRESULT = GetLastError();
        __leave;
    }
}
__finally {
    // Don't forget to free resources, if allocated.
    if (pCertContext != NULL) {
        CertFreeCertificateContext(pCertContext);
    }
    if (hCertStore != NULL) {
        CertCloseStore(hCertStore, CERT_CLOSE_STORE_FORCE_FLAG);
    }
}
return HRESULT;
}

```

12.5.3 使用私钥

从上节可知，在 Windows 环境下，智能卡作为密码设备，可以通过 CryptoAPI 或 CNG 接口操作。

12.1.3 节和 12.4.3 节都给出了使用私钥的例子，它们对智能卡也是适用的，可参考这些使用私钥的例子。

12.6 国密接口

12.6.1 国密接口简介

《智能 IC 卡及智能密码钥匙密码应用接口规范》简称为国密接口规范。此规范规定了基于 PKI 密码体制的智能 IC 卡及智能密码钥匙密码应用接口，描述了密码应用接口的函数、数据类型、参数的定义和设备的安全要求。

与 CryptoAPI、CNG 等相比，国密接口既支持 SM2 密码算法，又支持 RSA 算法，SM2 算法是国家密码局颁发的非对称密码算法。

智能 IC 卡及智能密码钥匙密码应用接口位于智能 IC 卡及智能密码钥匙应用程序与设备之间，如图 12-11 所示。

一个设备中存在设备认证密钥和多个应用，应用之间相互独立。设备的逻辑结构如图 12-12 所示。

应用由管理员 PIN、用户 PIN、文件和容器组成，可以存在多个文件和多个容器。每个应用维护各自的与管理员 PIN 和用户 PIN 相关的权限状态。应用的逻辑结构如图 12-13 所示。

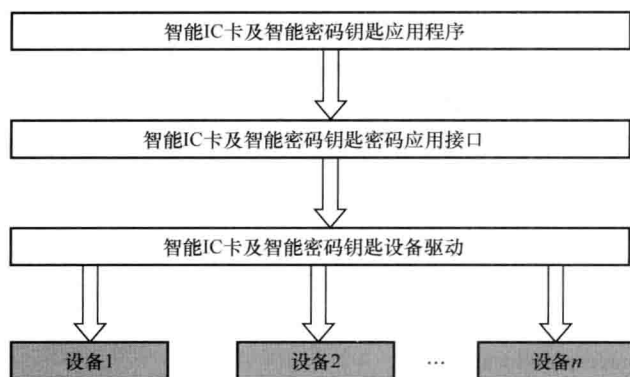


图 12-11 接口在应用层次关系中的位置

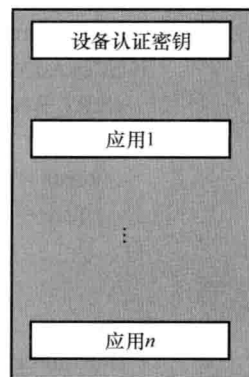


图 12-12 设备逻辑结构

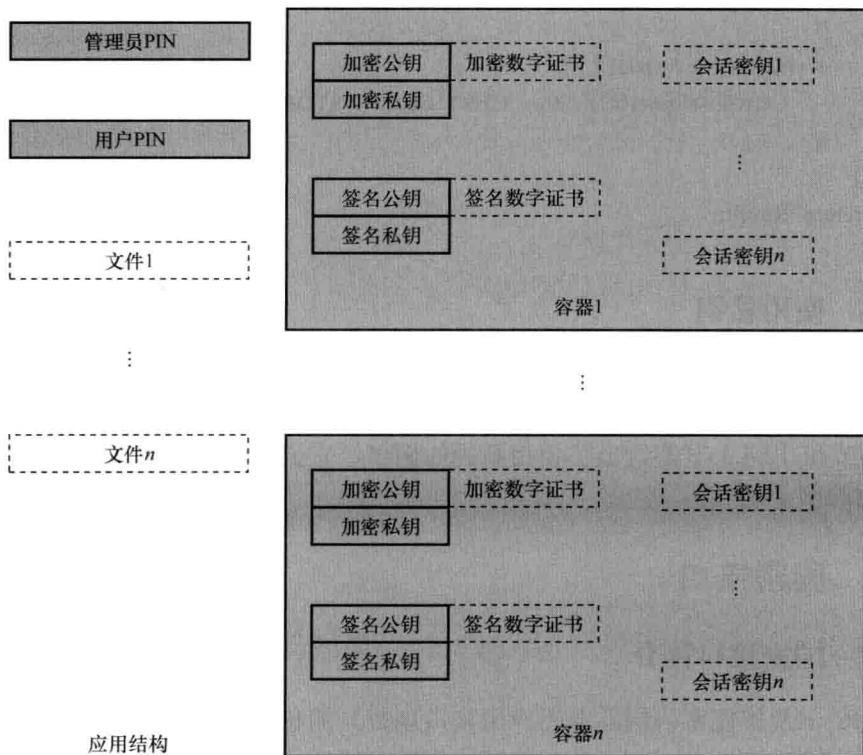


图 12-13 应用逻辑结构图

国密接口提供的接口函数分为“设备管理、访问控制、应用管理、文件管理、容器管理、密码服务”6个部分。

设备管理函数主要完成设备的插拔事件处理、枚举设备、连接设备、断开连接、获取设备状态、设置设备标签、获取设备信息、锁定设备、解锁设备和设备命令传输等操作。

访问控制函数主要完成设备认证、PIN码管理和安全状态管理等操作。

应用管理函数主要完成应用的创建、枚举、删除、打开、关闭等操作。

文件管理函数用以满足用户扩展开发的需要，包括创建文件、删除文件、枚举文件、获取文件信息、文件读写等操作。

容器管理函数包括创建、删除、枚举、打开和关闭容器的操作。

密码服务函数提供对称算法运算、非对称算法运算、密码哈希运算、密钥管理、消息鉴别码计算等功能。

12.6.2 使用证书

12.6.2.1 函数说明

1. 连接设备

函数原型	ULONG DEVAPI SKF_ConnectDev (LPSTR szName, DEVHANDLE *phDev)	
功能描述	通过设备名称连接设备，返回设备的句柄	
参数	szName	[IN] 设备名称
	phDev	[OUT] 返回设备操作句柄
返回值	SAR_OK: 成功 其他: 错误码	

2. 获取设备信息

函数原型	ULONG DEVAPI SKF_GetDevInfo (DEVHANDLE hDev, DEVINFO *pDevInfo)	
功能描述	获取设备的一些特征信息，包括设备标签、厂商信息、支持的算法等	
参数	hDev	[IN] 连接设备时返回的设备句柄
	pDevInfo	[OUT] 返回设备信息
返回值	SAR_OK: 成功 其他: 错误码	

3. 打开应用

函数原型	ULONG DEVAPI SKF_OpenApplication(DEVHANDLE hDev, LPSTR szAppName, HAPPLICATION *phApplication)	
功能描述	打开指定的应用	
参数	hDev	[IN] 连接设备时返回的设备句柄
	szAppName	[IN] 应用名称
	phApplication	[OUT] 应用的句柄
返回值	SAR_OK: 成功 其他: 错误码	

4. 打开容器

函数原型	ULONG DEVAPI SKF_OpenContainer(HAPPLICATION hApplication,LPSTR szContainerName, HCONTAINER *phContainer)	
功能描述	获取容器句柄	
参数	hApplication	[IN] 应用句柄
	szContainerName	[IN] 容器的名称
	phContainer	[OUT] 返回所打开容器的句柄
返回值	SAR_OK: 成功 其他: 错误码	

5. 导出数字证书

函数原型	ULONG DEVAPI SKF_ExportCertificate(HCONTAINER hContainer, BOOL bSignFlag, BYTE* pbCert, ULONG *pulCertLen)	
功能描述	从容器内导出数字证书	
参数	hContainer	[IN] 容器句柄
	bSignFlag	[IN] TRUE 表示签名证书, FALSE 表示加密证书
	pbCert	[OUT] 指向证书内容缓冲区, 如果此参数为 NULL, pulCertLen 表示返回数据所需要缓冲区的长度; 如果此参数不为 NULL, 返回数字证书内容
	pulCertLen	[IN/OUT] 输入时表示 pbCert 缓冲区的长度, 输出时表示证书内容的长度
返回值	SAR_OK: 成功 其他: 错误码	

12.6.2.2 示例程序

应用程序在获取证书时, 必须经过如下步骤: ①连接设备; ②获取设备信息 (可选); ③打开应用程序句柄; ④打开容器句柄; ⑤导出证书。

导出容器中证书示例代码如下:

```

ULONG      rv = 0;
DEVHANDLE hDev = NULL;
DEVINFO    DevInfo;
HAPPLICATION hApplication = NULL;
LPSTR      szAppName="My-Application";
LPSTR      containerName = "test";
HANDLE     hCon = NULL;
BYTE       bufCert[4096];
ULONG      ulCertLen = sizeof(bufCert);
do {
    // 打开设备句柄
    rv = SKF_ConnectDev( "xxx 设备名称" , &hDev);
    if (rv) {
        printf("call SKF_ConnectDev error.\n");
        break;
    }
    // 获得设备的一些信息, 可以进行展示
    rv = SKF_GetDevInfo(hDev, &DevInfo);
    if (rv) {
        printf("call SKF_GetDevInfo error.\n");
        break;
    }
    // 根据名称打开应用句柄, 应用名称为 szAppName 中存储的 "My-Application"

```

```

rv = SKF_OpenApplication(hDev, szAppName, &hApplication);
if (rv) {
    printf("call SKF_OpenApplication error.\n");
    break;
}
// 打开应用中容器，容器名称为 test
rv = SKF_OpenContainer(hApplication, "test", &hCon);
if (rv) {
    printf("call SKF_OpenContainer error\n");
    break;
}
// 导出容器中的签名证书，证书存储空间要预先分配
rv = SKF_ExportCertificate(hCon, TRUE, bufCert, &ulCertLen);
if (rv) {
    printf("call SKF_ExportCertificate error\n");
    break;
}
} while(0);
// 关闭打开句柄，以防止资源泄露
if (hCon) { SKF_CloseContainer(hCon); }
if (hApplication) { rv = SKF_CloseApplication(hApplication); }
if (hDev) { SKF_DisconnectDev(hDev); }

```

由于国密接口中没提供证书操作的接口函数，需要解析证书、获取证书中信息、验证证书有效性等功能，需要其他开发包，例如使用 CryptoAPI、OpenSSL 等。

12.6.3 使用私钥

12.6.3.1 函数说明

1. 校验 PIN

函数原型	ULONG WINAPI SKF_VerifyPIN (HAPPLICATION hApplication, ULONG ulPINType, LPSTR szPIN, ULONG *pulRetryCount)	
功能描述	校验 PIN 码。校验成功后，会获得相应的权限，如果 PIN 码错误，会返回 PIN 码的重试次数，当重试次数为 0 时表示 PIN 码已经锁死	
参数	hApplication	[IN] 应用句柄
	ulPINType	[IN] PIN 类型
	szPIN	[IN] PIN 值
	pulRetryCount	[OUT] 出错后返回的重试次数
返回值	SAR_OK: 成功 其他: 错误码	

2. ECC 签名

函数原型	ULONG DEVAPI SKF_ECCESignData (HCONTAINER hContainer, BYTE *pbData, ULONG ulDataLen, PECCSIGNATUREBLOB pSignature)	
功能描述	ECC 数字签名。采用 ECC 算法和指定私钥 hKey，对指定数据 pbData 进行数字签名。签名后的结果存放到 pSignature 中	
参数	hContainer	[IN] 密钥容器句柄
	pbData	[IN] 待签名的数据
	ulDataLen	[IN] 待签名数据长度，必须小于密钥模长
	pSignature	[OUT] 签名值
返回值	SAR_OK: 成功 其他: 错误码	
备注	权限要求：需要用户权限 输入数据为待签数据的哈希值	

3. 密码哈希初始化

函数原型	ULONG DEVAPI SKF_DigestInit(DEVHANDLE hDev, ULONG ulAlgID, ECCPUBLICKEYBLOB *pPubKey, unsigned char *pucID, ULONG ulIDLen, HANDLE *phHash)	
功能描述	初始化密码哈希计算操作，指定计算密码哈希值的算法	
参数	hDev	[IN] 连接设备时返回的设备句柄
	ulAlgID	[IN] 密码哈希算法标识
	pPubKey	[IN] 签名者公钥。当 ulAlgID 为 SGD_SM3 时有效
	pucID	[IN] 签名者的 ID 值，当 ulAlgID 为 SGD_SM3 时有效
	ulIDLen	[IN] 签名者 ID 的长度，当 ulAlgID 为 SGD_SM3 时有效
	phHash	[OUT] 密码哈希对象句柄
返回值	SAR_OK: 成功 其他: 错误码	
备注	在 ulAlgID 为 SGD_SM3 且 ulIDLen 不为 0 的情况下 pPubKey、pucID 有效，执行 SM2 算法签名预处理 1 操作（具体参见“8.2 SM2”节内容）	

4. 多组数据密码哈希

函数原型	ULONG DEVAPI SKF_DigestUpdate (HANDLE hHash, BYTE *pbData, ULONG ulDataLen)	
功能描述	对多个分组的消息进行密码哈希计算。调用 SKF_DigestUpdate 之前，必须调用 SKF_DigestInit 初始化密码哈希计算操作；调用 SKF_DigestUpdate 之后，必须调用 SKF_DigestFinal 结束密码哈希计算操作	
参数	hHash	[IN] 密码哈希对象句柄
	pbData	[IN] 指向消息数据的缓冲区
	ulDataLen	[IN] 消息数据的长度
返回值	SAR_OK: 成功 其他: 错误码	

5. 结束密码哈希

函数原型	ULONG WINAPI SKF_DigestFinal (HANDLE hHash, BYTE *pHashData, ULONG *pulHashLen)	
功能描述	结束多个分组消息的密码哈希计算操作，将密码哈希结果保存到指定的缓冲区	
参数	hHash	[IN] 密码哈希对象句柄
	pHashData	[OUT] 返回的密码哈希结果缓冲区指针，如果此参数为 NULL，则由 pulHashLen 返回哈希结果的长度
	pulHashLen	[IN, OUT] 输入时表示哈希结果缓冲区的长度，输出时表示密码哈希结果的长度
返回值	SAR_OK: 成功 其他: 错误码	
备注	SKF_DigestFinal 必须用于 SKF_DigestUpdate 之后	

6. 数据结构

(1) 公钥数据结构 ECCPUBLICKEYBLOB

```
typedef struct Struct_ECCPUBLICKEYBLOB {
    ULONG    BitLen;
    BYTE     XCoordinate[ECC_MAX_XCOORDINATE_BITS_LEN/8];
    BYTE     YCoordinate[ECC_MAX_YCOORDINATE_BITS_LEN/8];
} ECCPUBLICKEYBLOB, *PECCPUBLICKEYBLOB;
```

其中，BitLen：模数的实际位长度，必须是 8 的倍数。XCoordinate：曲线上点的 X 坐标，有限域上的整数。YCoordinate：曲线上点的 Y 坐标，有限域上的整数。

```
#define ECC_MAX_XCOORDINATE_BITS_LEN 512
#define ECC_MAX_YCOORDINATE_BITS_LEN 512
```

ECC_MAX_XCOORDINATE_LEN 与 ECC_MAX_YCOORDINATE_LEN 分别为 ECC 算法 X 坐标和 Y 坐标的最大长度。

(2) 签名数据结构 ECCSIGNATUREBLOB

```
typedef struct Struct_ECCSIGNATUREBLOB {
    BYTE r[ECC_MAX_XCOORDINATE_BITS_LEN/8];
    BYTE s[ECC_MAX_XCOORDINATE_BITS_LEN/8];
} ECCSIGNATUREBLOB, *PECCSIGNATUREBLOB;
```

其中，r：签名结果的 r 部分；s：签名结果的 s 部分。

```
#define ECC_MAX_MODULUS_BITS_LEN 512
```

ECC_MAX_MODULUS_BITS_LEN 为 ECC 算法模数的最大长度。

12.6.3.2 示例程序

与访问证书相比，使用私钥时，必须进行身份认证，即在打开应用句柄后，必须进行 PIN 码认证。其步骤包括：①连接设备；②获取设备信息（可选）；③打开应用句柄；④进行 PIN 码认证；⑤打开容器句柄；⑥进行签名。

使用私钥进行签名的示例代码如下：

```

ULONG      rv = 0;
DEVHANDLE hDev = NULL;
DEVINFO    DevInfo;
HAPPLICATION hApplication = NULL;
LPSTR      szAppName="My-Application";
LPSTR      containerName = "test";
HANDLE     hCon = NULL;
BYTE       bufCert[4096];
ULONG      ulCertLen = sizeof(bufCert);

LPSTR      szUserPin = "111111";
LPSTR      szUserId = "1234567812345678";
DWORD      dwUserPinRetryCount=10;
ECCSIGNATUREBLOB pSignature;
ECCPUBLICKEYBLOB eccSignPubKey;
HANDLE     hDigest = NULL;
BYTE       pbData[3072] = "12345678901234567890", pbDigest[64] = {0};
ULONG      ulDataLen = 33, ulSig = 0, ulDigest, pukLen = 0;

do {
    // 打开设备句柄
    rv = SKF_ConnectDev( "xxx 设备名称" , &hDev);
    if (rv) {
        printf("call SKF_ConnectDev error.\n");
        break;
    }
    // 获得设备的一些信息，可以进行展示
    rv = SKF_GetDevInfo(hDev, &DevInfo);
    if (rv) {
        printf("call SKF_GetDevInfo error.\n");
        break;
    }
    // 根据名称打开应用句柄，应用名称为 szAppName 中存储的 “My-Application”
    rv = SKF_OpenApplication(hDev, szAppName, &hApplication);
    if (rv) {
        printf("call SKF_OpenApplication error.\n");
        break;
    }
    // 验证用户 PIN 码
    rv = SKF_VerifyPIN(hApplication, USER_TYPE, szUserPin, &dwUserPinRetryCount);
    printf("call SKF_VerifyPIN error.\n");
    break;
}

```

```

// 打开应用中容器，容器名称为 test
rv = SKF_OpenContainer(hApplication, "test", &hCon);
if (rv) {
    printf("call SKF_OpenContainer error\n");
    break;
}
// 导出容器中的签名公钥，用于计算签名哈希值
// TRUE 导出签名密钥，FALSE 导出加密密钥
pukLen = sizeof(eccSignPubKey);
rv = SKF_ExportPublicKey(hCon, TRUE, (unsigned char*)&eccSignPubKey, &pukLen);
if (rv) {
    printf("SKF_ExportPublicKey(%0x) failed\n", rv);
    break;
}
// 计算签名哈希值
rv = SKF_DigestInit(hDev, SGD_SM3, &eccSignPubKey, szUserId, 16, &hDigest);
if (rv) {
    printf("call SKF_DigestInit error(%0x)\n", rv);
    break;
}
SKF_DigestUpdate(hDigest, pbData, totalLen);
ulDigest = sizeof(pbDigest);
rv = SKF_DigestFinal(hDigest, pbDigest, &ulDigest);
if (rv) {
    printf("call SKF_Digest error(%0x)\n", rv);
    break;
}
SKF_CloseHandle(hDigest);
hDigest = NULL;
// 进行签名
ulSig = sizeof(pSignature);
rv = skf_ECCECCSignData(hCon, pbDigest, ulDigest, &pSignature);
if (rv) {
    printf("Error: Sign Len(%ld) - result(%0x)\n", ulDataLen, rv);
    break;
}
} while(0);
// 关闭打开句柄，以防止资源泄露
if (hDigest) { SKF_CloseHandle(hDigest); }
if (hCon) { SKF_CloseContainer(hCon); }
if (hApplication) { rv = SKF_CloseApplication(hApplication); }
if (hDev) { SKF_DisconnectDev(hDev); }

```

在计算 SM2 签名值前，需要对数据进行哈希运算，此哈希运算需要签名者公钥和用户 ID 参数。

第 13 章 实 验 二

13.1 RSA 公钥格式编码示例

13.1.1 ASN.1 描述与实例

1. ASN.1 描述

RSA 公钥格式用 ASN.1 描述如下：

```
RSAPublicKey ::= SEQUENCE {  
    modulus          INTEGER, -- n  
    publicExponent   INTEGER  -- e  
}
```

2. RSA 公钥实例

假设 RSA 公钥 $PK=\{e, n\}$ ， e 为 65537， n 为 128 字节的大整数，用十六进制表示为：

```
n=B4 F6 CF 18 3D 5E 8E 1D 46 7A 90 7D 8E 41 D2 E3  
C8 F1 A3 AE F3 6D 8A 24 FF 55 23 25 BD EB 0C D0  
7B 87 36 5D 1F 73 98 65 3E 57 97 F6 65 7D 13 E0  
E1 B5 FC BC 38 6F 56 3E 57 4E D6 51 1D 13 12 7C  
33 B3 60 31 79 32 07 97 F3 3C 8B 29 0D B5 78 38  
93 CE 84 E4 A3 DD FB F9 25 47 1C 72 A6 5E 78 02  
CF F3 48 9D CA D9 00 73 DE 4B 16 07 52 48 20 06  
F3 4F CA A5 2D 66 88 95 C6 6C D6 3F 61 34 F7 E3  
（简写为：B4 F6...F7 F3）  
e=01 00 01
```

13.1.2 DER 编码过程

1. 对 e 和 n 进行 DER 编码

e 和 n 为 INTEGER 结构类型，编码规则采用基本类型定长模式。

对于标识串，采用低标识编码方式，只需 1 个字节。INTEGER 的 tag 为 0x02；class 选择 universal，则位 8 和位 7 为 0，INTEGER 为基本类型，则位 6 为 0。因此，标识串=0x02。

对于长度串， e 采用短型编码方式，只需 1 个字节， n 采用长型编码方式，需要 2 个字节。

对于内容串，由 e 和 n 的十六进制值组成。由于 INTEGER 类型 DER 编码后第 1 字节位 8 表示正负整数，因此如果正整数第 1 字节位 8 为 1 时，在前填充 1 个字节 0x00。

具体编码过程如表 13-1 所示。

表 13-1 RSA 公钥参数 e 和 n 编码过程

RSA 公钥参数	标识串	长度串	内容串
n=B4 F6…F7 F3	02	81 81	00 B4 F6…F7 F3
e=01 00 01	02	03	01 00 01

2. 对 RSAPublicKey 进行 DER 编码

RSAPublicKey 为 SEQUENCE 结构类型，编码规则采用结构类型定长模式。

对于标识串，采用低标识编码方式，只需 1 个字节。SEQUENCE 的 tag 为 0x10；class 选择 universal，则位 8 和位 7 为 0，SEQUENCE 为结构类型，则位 6 为 1。因此，标识串=0x30。

对于长度串，采用长型编码方式，需要 2 个字节。

对于内容串，由 modulus 和 publicExponent 的 DER 编码值组成。

具体编码过程如表 13-2 所示。

表 13-2 RSAPublicKey 编码过程

RSA 公钥	标识串	长度串	内容串
RSAPublicKey	30	81 89	02 81 81 00 B4 F6…F7 F3 02 03 01 00 01

13.2 数字证书格式编码示例

13.2.1 ASN.1 描述与实例

以第 11 章中 ZHANG San 的数字证书为例，序列号=1174 (0x0496)，证书签发者 DN=“CN = Virtual CA, C = CN”，证书持有者 DN=“CN = ZHANG San, OU = Person, C = CN”，证书有效期=20140222000000-20160222000000。

1. TBSCertificate 的 ASN.1 描述与实例

TBSCertificate 格式用 ASN.1 描述如下：

```
TBSCertificate ::= SEQUENCE {
    version          [0] EXPLICIT Version DEFAULT v1,
    serialNumber      CertificateSerialNumber,
    signature         AlgorithmIdentifier,
    issuer            Name,
    validity          Validity,
    subject           Name,
    subjectPublicKeyInfo SubjectPublicKeyInfo,
    issuerUniqueID    [1] IMPLICIT UniqueIdentifier OPTIONAL,
                      -- If present, version MUST be v2 or v3
```

```

subjectUniqueID [2]  IMPLICIT UniqueIdentifier OPTIONAL,
                        -- If present, version MUST be v2 or v3
extensions        [3]  EXPLICIT Extensions OPTIONAL
                        -- If present, version MUST be v3
}

Extensions ::= SEQUENCE SIZE (1..MAX) OF Extension

```

TBSCertificate 中各项内容具体值如表 13-3 所示。

表 13-3 TBSCertificate 内容值

TBSCertificate	值
version	02 (十六进制)
serialNumber	04 96 (十六进制)
signature	sha1WithRSAEncryption (1.2.840.113549.1.1.5)
issuer	“CN=Virtual CA, C=CN”
validity	notBefore=20140222000000、notAfter=20160222000000
subject	“CN=ZHANG San, OU=Person, C=CN”
subjectPublicKeyInfo	同 13.1 节
issuerUniqueID subjectUniqueID	空
extensions	包含 6 个扩展项 (Extension): basicConstraints、subjectKeyIdentifier、keyUsage、extKeyUsage、netscapeCertType、authorityKeyIdentifier

2. Extension 的 ASN.1 描述与实例

Extension 格式用 ASN.1 描述如下:

```

Extension ::= SEQUENCE {
    extnID      OBJECT IDENTIFIER,
    critical    BOOLEAN DEFAULT FALSE,
    extnValue   OCTET STRING }

```

Extension 各扩展项值如表 13-4 所示。

表 13-4 Extension 各扩展项值

Extension	值
basicConstraints	关键项, 值为空 (表示 cA=FALSE)
subjectKeyIdentifier	2C 04 87 10 60 FC 61 F6 2B 64 81 3D FB 66 30 DA F0 73 BC 08 (SHA1 值, 简写为: 2C 04...BC 08)
keyUsage	关键项, 值为 Digital Signature、Non-Repudiation、Key Encipherment、Data Encipherment、Key Agreement (F8)
extKeyUsage	客户端身份验证 (1.3.6.1.5.5.7.3.2)、智能卡登录 (1.3.6.1.4.1.311.20.2.2)、安全电子邮件 (1.3.6.1.5.5.7.3.4)
netscapeCertType	SSL 客户端身份验证、SMIME (A0)
authorityKeyIdentifier	96 F0 94 F8 49 8D 23 05 86 B0 CA B5 2D 7A 9A 60 32 FB B0 F9 (简写为: 96 F0...B0 F9)

3. Certificate 的 ASN.1 描述与实例

Certificate 格式用 ASN.1 描述如下：

```
Certificate ::= SEQUENCE {
    tbsCertificate          TBSCertificate,
    signatureAlgorithm      AlgorithmIdentifier,
    signatureValue          BIT STRING }
```

Certificate 中各项内容的具体值如表 13-5 所示。

表 13-5 Certificate 内容值

Certificate	值
tbsCertificate	见 DER 编码过程
signatureAlgorithm	sha1WithRSAEncryption (1.2.840.113549.1.1.5)
signatureValue	8D 42 AD 5C DF C7 C7 90 FA 58 C0 74 15 C6 4F 20 9B F1 49 9C B8 3C 22 98 45 75 A6 0D 7C 02 9D 83 1D C4 5D CF 4F 8E 57 E7 0A 9B 67 02 33 23 59 76 B4 B5 B7 F3 27 36 6F F4 32 6C 1C E9 B3 4B 81 DC D0 CF 2E CF 07 4C 65 75 74 DF 23 9D 7D 2B E4 F1 15 0C 84 61 41 5F DC 67 92 A9 7C 39 A0 CA A9 58 6B ED 7D 94 08 F7 83 42 61 F8 62 D8 DC 3B 5D B7 69 5C D0 36 F2 99 A8 0C 99 6E B0 0C 21 E3 98 9F 12 6D D1 76 4E 0C 31 CB 7F 54 73 FE 96 83 76 35 22 2F BF F6 2B 11 04 3A A7 BE 33 3C D5 DA EE 56 7A C4 1A 67 3B 77 DE 52 C0 DA 09 CA 45 71 11 B2 D5 35 BF 44 54 08 C2 FA 0C 5C EF C0 EF 82 63 37 3C 4C AB 59 4C FD 6C 2A 9D 64 27 35 4E 4F D8 2E 2C 5C EB A1 99 DB FA 3A 53 54 13 92 91 5D 8F 38 DD 1C D8 AB 34 22 9A EF 8A E4 62 C2 23 9D 06 A5 D7 D8 58 B7 F4 98 CA 61 29 9D DE A8 F6 DA CC 81 (256 字节, 简写为: 8D 42...CC 81)

13.2.2 DER 编码过程

1. 对 Extension 进行 DER 编码

各扩展项具体内容用 ASN.1 描述如下：

```
BasicConstraints ::= SEQUENCE {
    cA                      BOOLEAN DEFAULT FALSE,
    pathLenConstraint       INTEGER (0..MAX) OPTIONAL }
SubjectKeyIdentifier ::= KeyIdentifier
    (KeyIdentifier ::= OCTET STRING)
KeyUsage ::= BIT STRING
ExtKeyUsageSyntax ::= SEQUENCE SIZE (1..MAX) OF KeyPurposeId
```

```

(KeyPurposeId ::= OBJECT IDENTIFIER)
NetscapeCertType ::= BIT STRING
AuthorityKeyIdentifier ::= SEQUENCE {
    keyIdentifier          [0] KeyIdentifier          OPTIONAL,
    authorityCertIssuer    [1] GeneralNames          OPTIONAL,
    authorityCertSerialNumber [2] CertificateSerialNumber OPTIONAL }
(KeyIdentifier ::= OCTET STRING)

```

Extension 为 SEQUENCE 结构类型，不同扩展项 DER 编码值包含在 OCTET STRING 类型 extnValue 中，编码规则采用结构类型定长模式。各扩展项 DER 编码值用括号分隔。其中，对于 BIT STRING 类型，编码后第 1 个字节表示填充位数或未使用位数。

Extension 具体编码过程如表 13-6 所示。

表 13-6 Extension 编码过程

Extension	标识串	长度串	内 容 串
basicConstraints	30	0C	06 03 55 1D 13 01 01 FF 04 02 (30 00)
subjectKeyIdentifier	30	1D	06 03 55 1D 0E 04 16 (04 14 2C 04...BC 08)
keyUsage	30	0E	06 03 55 1D 0F 01 01 FF 04 04 (03 02 03 F8)
extKeyUsage	30	29	06 03 55 1D 25 04 22 (30 20 06 08 2B 06 01 05 05 07 03 02 06 0A 2B 06 01 04 01 82 37 14 02 02 06 08 2B 06 01 05 05 07 03 04)
netscapeCertType	30	11	06 09 60 86 48 01 86 F8 42 01 01 04 04 (03 02 05 A0)
authorityKeyIdentifier	30	1F	06 03 55 1D 23 04 18 (30 16 80 14 96 F0...B0 F9)

2. 对 TBSCertificate 进行 DER 编码

TBSCertificate 内容编码规则采用结构类型定长模式，具体编码过程如表 13-7 所示。

表 13-7 TBSCertificate 内容编码过程

TBSCertificate	标识串	长度串	内 容 串
version [0] EXPLICIT	A0	03	02 01 02
serialNumber	02	02	04 96
signature	30	0D	06 09 2A 86 48 86 F7 0D 01 01 05 05 00

(续表)

TBSCertificate	标识串	长度串	内 容 串
issuer	30	22	31 0B 30 09 06 03 55 04 06 13 02 43 4E 31 13 30 11 06 03 55 04 03 13 0A 56 69 72 74 75 61 6C 20 43 41
validity	30	1E	17 0D 31 34 30 32 32 31 31 36 30 30 30 30 5A 17 0D 31 36 30 32 32 31 31 36 30 30 30 30 5A
subject	30	32	31 0B 30 09 06 03 55 04 06 13 02 43 4E 31 0F 30 0D 06 03 55 04 0B 13 06 50 65 72 73 6F 6E 31 12 30 10 06 03 55 04 03 13 09 5A 48 41 4E 47 20 53 61 6E
subjectPublicKeyInfo	30	81 9F	30 0D 06 09 2A 86 48 86 F7 0D 01 01 01 05 00 03 81 8D 00 30 81 89 02 81 81 00 B4 F6 ... F7 E3 02 03 01 00 01
extensions [3] EXPLICIT	A3	81 9F	30 81 9C 30 0C ...30 00: basicConstraints 30 1D...BC 08: subjectKeyIdentifier 30 0E...03 F8: keyUsage 30 29...03 04: extKeyUsage 30 11...05 A0: netscapeCertType 30 1F...b0 F9: authorityKeyIdentifier

TBSCertificate 为 SEQUENCE 结构类型，编码规则采用结构类型定长模式。
TBSCertificate 具体编码过程如表 13-8 所示。

表 13-8 TBSCertificate 编码过程

TBSCertificate	标识串	长度串	内 容 串
TBSCertificate	30	82 01 D4	A0 03...01 02: version 02 02...04 96: serialNumber 30 0D...05 00: signature 30 22...43 41: issuer 30 1E...30 5A: validity 30 32...61 6E: subject 30 81...00 01: subjectPublicKeyInfo A3 81...B0 F9: extensions

3. 对 Certificate 进行 DER 编码

Certificate 内容编码规则采用结构类型定长模式，具体编码过程如表 13-9 所示。

表 13-9 Certificate 内容编码过程

Certificate	标识串	长度串	内 容 串
tbsCertificate	30	82 01 D4	A0 03...B0 F9
signatureAlgorithm	30	0D	06 09 2A 86 48 86 F7 0D 01 01 05 05 00
signatureValue	03	82 01 01	00 8D 42...CC 81

Certificate 为 SEQUENCE 结构类型，编码规则采用结构类型定长模式。
Certificate 具体编码过程如表 13-10 所示。

表 13-10 Certificate 编码过程

Certificate	标识串	长度串	内 容 串
Certificate	30	82 02 EC	30 82...B0 F9: tbsCertificate 30 0D...05 00: signatureAlgorithm 03 82...CC 81: signatureValue

该数字证书 DER 编码后的文件大小为 752 字节，具体二进制值如表 13-11 所示。其中，每行显示 16 个字节，每行最前面 4 个数字表示该行第 1 个字节的地址序号（从 0 开始）。

表 13-11 数字证书 DER 文件内容

```

0000: 30 82 02 EC 30 82 01 D4 -- A0 03 02 01 02 02 02 04
0010: 96 30 0D 06 09 2A 86 48 -- 86 F7 0D 01 01 05 05 00
0020: 30 22 31 0B 30 09 06 03 -- 55 04 06 13 02 43 4E 31
0030: 13 30 11 06 03 55 04 03 -- 13 0A 56 69 72 74 75 61
0040: 6C 20 43 41 30 1E 17 0D -- 31 34 30 32 32 31 31 36
0050: 30 30 30 30 5A 17 0D 31 -- 36 30 32 32 31 31 36 30
0060: 30 30 30 5A 30 32 31 0B -- 30 09 06 03 55 04 06 13
0070: 02 43 4E 31 0F 30 0D 06 -- 03 55 04 0B 13 06 50 65
0080: 72 73 6F 6E 31 12 30 10 -- 06 03 55 04 03 13 09 5A
0090: 48 41 4E 47 20 53 61 6E -- 30 81 9F 30 0D 06 09 2A
00A0: 86 48 86 F7 0D 01 01 01 -- 05 00 03 81 8D 00 30 81
00B0: 89 02 81 81 00 B4 F6 CF -- 18 3D 5E 8E 1D 46 7A 90
00C0: 7D 8E 41 D2 E3 C8 F1 A3 -- AE F3 6D 8A 24 FF 55 23
00D0: 25 BD EB 0C D0 7B 87 36 -- 5D 1F 73 98 65 3E 57 97
00E0: F6 65 7D 13 E0 E1 B5 FC -- BC 38 6F 56 3E 57 4E D6
00F0: 51 1D 13 12 7C 33 B3 60 -- 31 79 32 07 97 F3 3C 8B
0100: 29 0D B5 78 38 93 CE 84 -- E4 A3 DD FB F9 25 47 1C
0110: 72 A6 5E 78 02 CF F3 48 -- 9D CA D9 00 73 DE 4B 16
0120: 07 52 48 20 06 F3 4F CA -- A5 2D 66 88 95 C6 6C D6
0130: 3F 61 34 F7 E3 02 03 01 -- 00 01 A3 81 9F 30 81 9C
0140: 30 0C 06 03 55 1D 13 01 -- 01 FF 04 02 30 00 30 1D
0150: 06 03 55 1D 0E 04 16 04 -- 14 2C 04 87 10 60 FC 61
0160: F6 2B 64 81 3D FB 66 30 -- DA F0 73 BC 08 30 0E 06
0170: 03 55 1D 0F 01 01 FF 04 -- 04 03 02 03 F8 30 29 06
0180: 03 55 1D 25 04 22 30 20 -- 06 08 2B 06 01 05 05 07
0190: 03 02 06 0A 2B 06 01 04 -- 01 82 37 14 02 02 06 08
01A0: 2B 06 01 05 05 07 03 04 -- 30 11 06 09 60 86 48 01
01B0: 86 F8 42 01 01 04 04 03 -- 02 05 A0 30 1F 06 03 55
01C0: 1D 23 04 18 30 16 80 14 -- 96 F0 94 F8 49 8D 23 05
01D0: 86 B0 CA B5 2D 7A 9A 60 -- 32 FB B0 F9 30 0D 06 09
01E0: 2A 86 48 86 F7 0D 01 01 -- 05 05 00 03 82 01 01 00
01F0: 8D 42 AD 5C DF C7 C7 90 -- FA 58 C0 74 15 C6 4F 20
0200: 9B F1 49 9C B8 3C 22 98 -- 45 75 A6 0D 7C 02 9D 83
0210: 1D C4 5D CF 4F 8E 57 E7 -- 0A 9B 67 02 33 23 59 76

```

(续表)

```

0220: B4 B5 B7 F3 27 36 6F F4 -- 32 6C 1C E9 B3 4B 81 DC
0230: D0 CF 2E CF 07 4C 65 75 -- 74 DF 23 9D 7D 2B E4 F1
0240: 15 0C 84 61 41 5F DC 67 -- 92 A9 7C 39 A0 CA A9 58
0250: 6B ED 7D 94 08 F7 83 42 -- 61 F8 62 D8 DC 3B 5D B7
0260: 69 5C D0 36 F2 99 A8 0C -- 99 6E B0 0C 21 E3 98 9F
0270: 12 6D D1 76 4E 0C 31 CB -- 7F 54 73 FE 96 83 76 35
0280: 22 2F BF F6 2B 11 04 3A -- A7 BE 33 3C D5 DA EE 56
0290: 7A C4 1A 67 3B 77 DE 52 -- C0 DA 09 CA 45 71 11 B2
02A0: D5 35 BF 44 54 08 C2 FA -- 0C 5C EF C0 EF 82 63 37
02B0: 3C 4C AB 59 4C FD 6C 2A -- 9D 64 27 35 4E 4F D8 2E
02C0: 2C 5C EB A1 99 DB FA 3A -- 53 54 13 92 91 5D 8F 38
02D0: DD 1C D8 AB 34 22 9A EF -- 8A E4 62 C2 23 9D 06 A5
02E0: D7 D8 58 B7 F4 98 CA 61 -- 29 9D DE A8 F6 DA CC 81

```

13.3 Windows 证书库操作示例

13.3.1 查看证书库内容

1. 进入证书库

打开 IE 浏览器，单击菜单“Internet 选项”后进入“内容”页，单击“证书”按钮即可进入证书库界面，如图 13-1 所示。

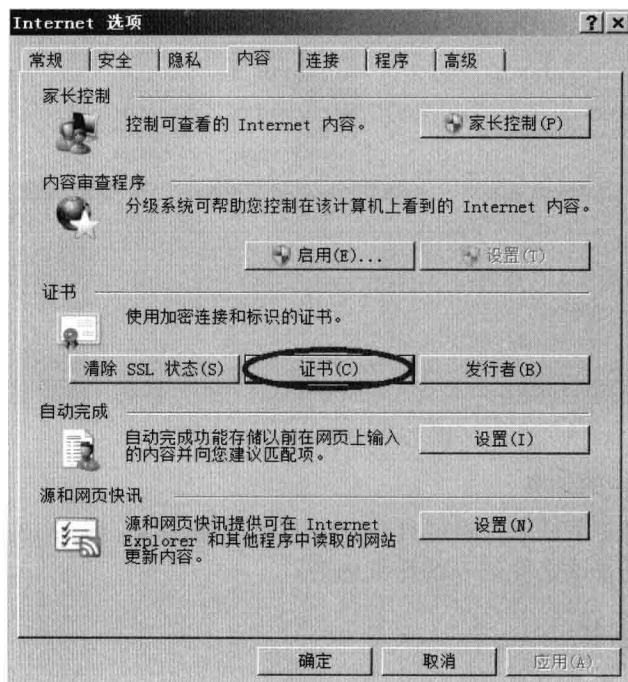


图 13-1 IE 浏览器 Internet 选项

2. 查看证书库

Windows 证书库分为以下几类：

(1) 受信任的根证书颁发机构

保存多个可信任的根 CA 证书。通过该类证书可以验证用户证书或子 CA 证书的合法性。

(2) 中级证书颁发机构

保存多个子 CA 证书。

(3) 受信任的发布者

保存多个可信任的可执行代码发布者证书。如果某可执行程序具有代码签名，且使用该证书能验证代码签名的合法性，则说明该程序值得信赖。

(4) 未受信任的发布者

保存多个不受信任的可执行代码发布者证书。如果某可执行程序具有代码签名，且使用该证书能验证代码签名的合法性，则说明该程序不受信任，可能存在安全风险，不建议安装或使用。

(5) 其他人证书

保存多个他人的证书。

(6) 个人证书

包含自己的数字证书，如果有对应的私钥，则与其关联。

单击不同的标签页即可进入不同的证书库，如图 13-2 所示。

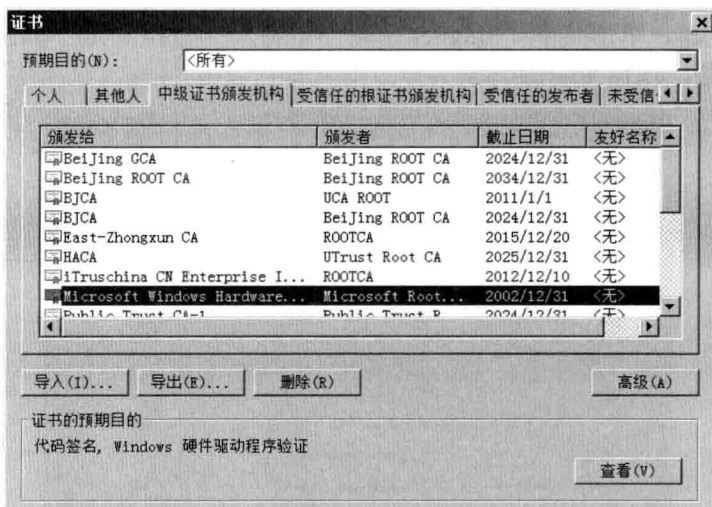


图 13-2 Windows 证书库

3. 查看证书库中的证书

双击证书库中的某个证书，即可查看该证书的详细内容，如图 13-3 所示。双击某个证书文件名称，也可查看该文件证书的详细内容。

13.3.2 导入证书

进入证书导入向导有两种方法。

方法一：在证书库查看界面中（见图 13-2），单击“导入”按钮后进入证书导入向导。

方法二：双击某个证书文件名称，将进入该证书详细内容查看界面，单击“安装证书”按钮后进入证书导入向导，如图 13-4 所示。

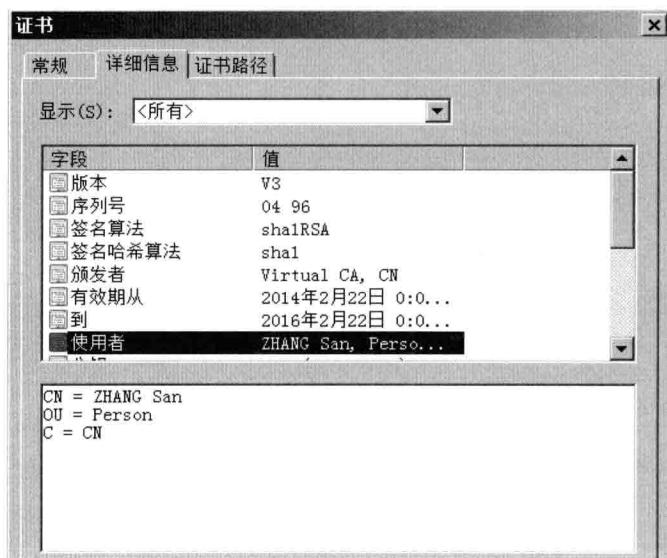


图 13-3 数字证书详细内容查看

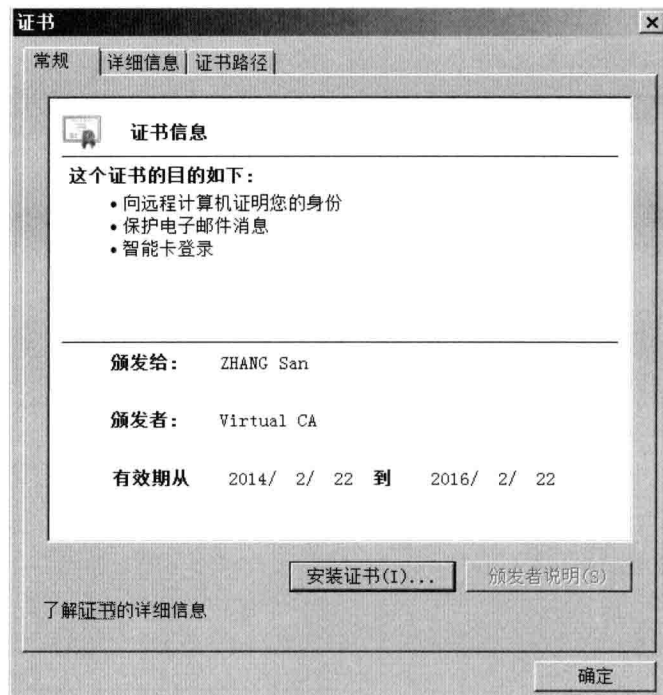


图 13-4 通过查看文件证书进入证书导入向导

首先，选择需要导入的证书文件名称，如图 13-5 所示。支持的证书文件类型包括 PKCS#12 (P12 或 PFX)、PKCS#7 (P7B)、CER 或 DER 等。

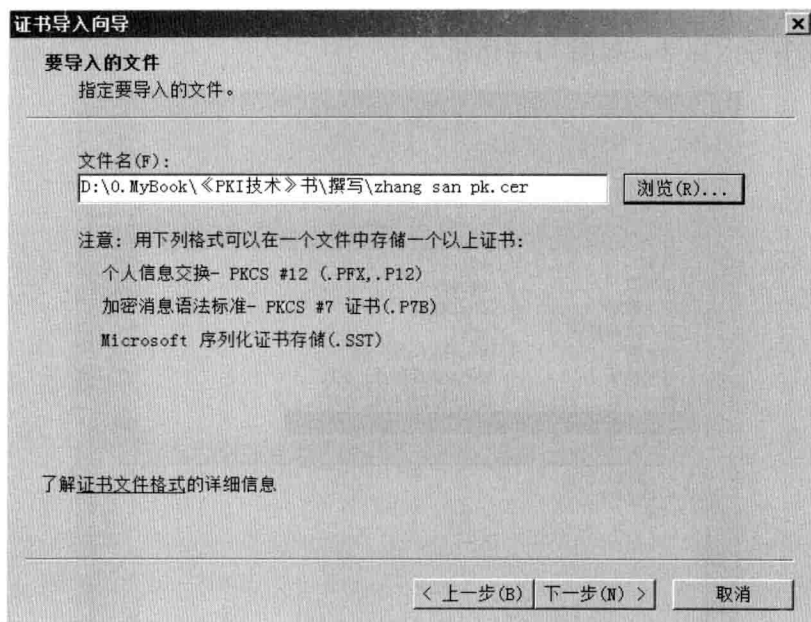


图 13-5 导入证书时证书文件名称选择

然后，设置证书存储位置选择方式，如图 13-6 所示。支持两种选择方式：根据证书类型自动选择和手工选择。

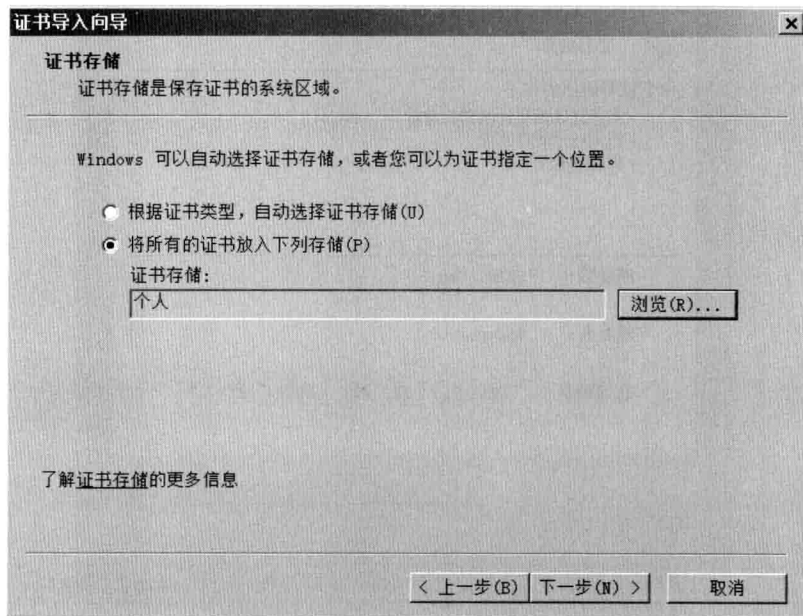


图 13-6 导入证书时证书存储位置选择方式设置

单击“浏览”按钮后，可手工选择证书存储位置，如图 13-7 所示。

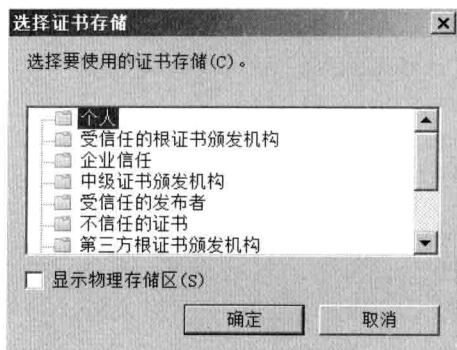


图 13-7 导入证书时手工选择证书存储位置

按照提示依次单击“下一步”按钮，将会成功导入证书到 Windows 证书库。

13.3.3 导出证书

在证书库查看界面中（见图 13-2），选择某证书后单击“导出”按钮，即进入证书导出向导。

首先，选择“您想将私钥跟证书一起导出吗？”，如图 13-8 所示。

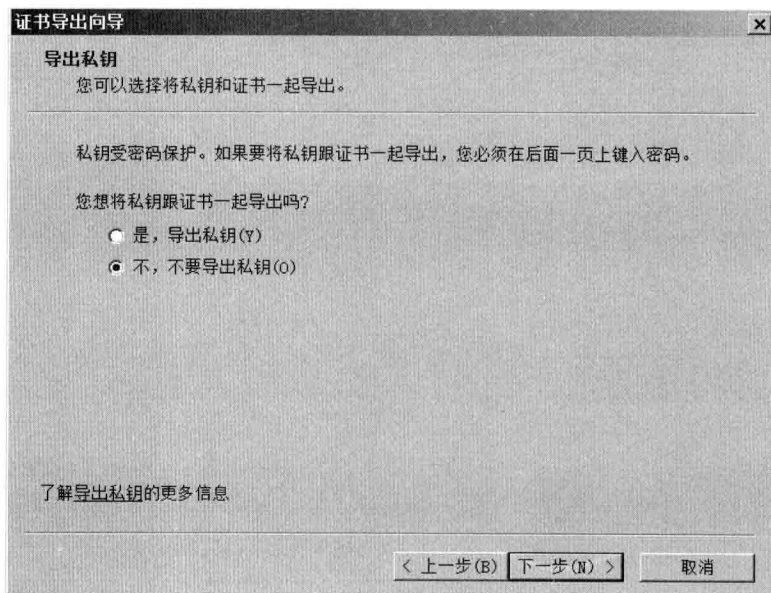


图 13-8 导出证书时选择是否导出私钥

然后，选择导出文件格式，如图 13-9 所示。支持的证书文件格式包括 PKCS#12（P12 或 PFX）、PKCS#7（P7B）、DER 编码二进制 X.509（CER）、Base64 编码 X.509（CER，文本格式）等。

最后，选择需要保存的证书文件名称，如图 13-10 所示。

按照提示依次单击“下一步”按钮，将会成功从 Windows 证书库导出指定证书。

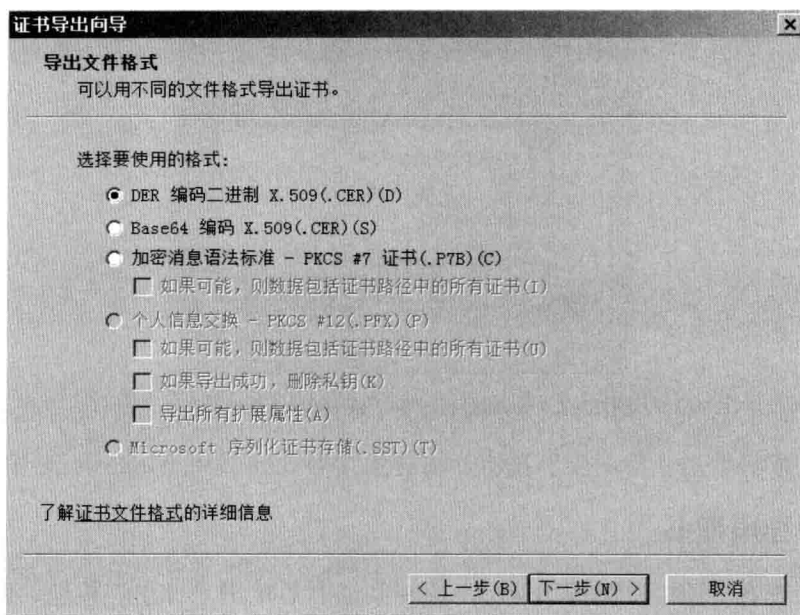


图 13-9 导出证书时选择文件格式

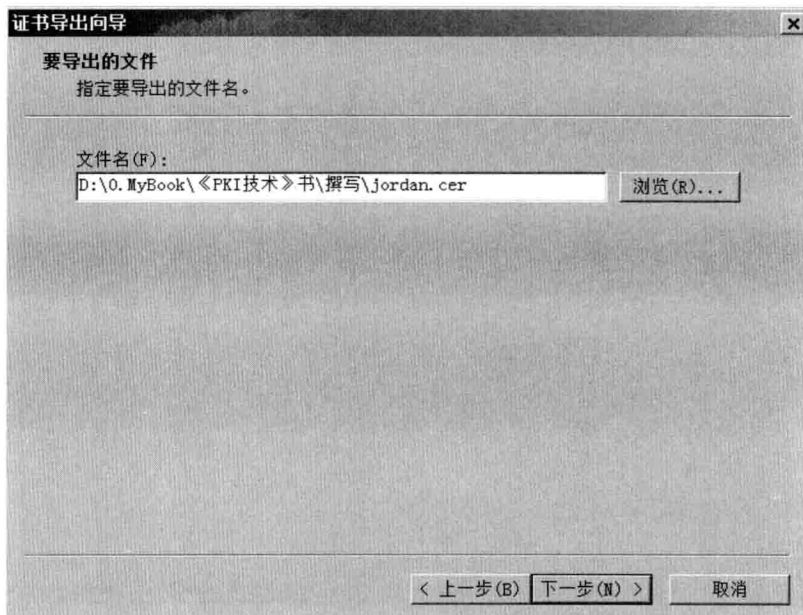


图 13-10 导出证书时选择证书文件名称

第四部分

PKI 之 CA 与 KMC: 管理网络身份证

第14章 系统结构

14.1 国际标准

IETF RFC 3280 规定了 CA 的系统结构，其相关实体及其关系如图 14-1 所示。

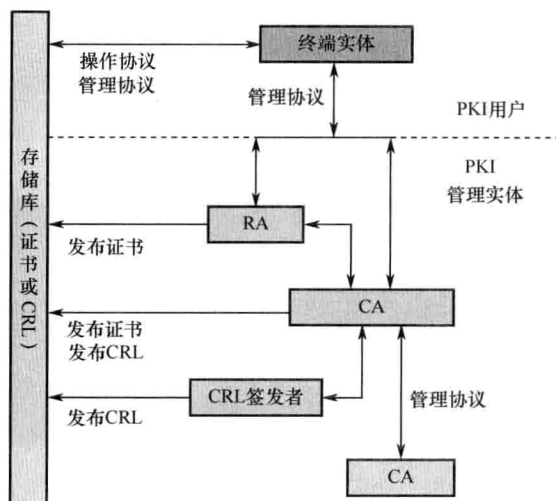


图 14-1 CA 相关实体及其关系

1. CA 相关实体（Entities）

① 终端实体（End Entity）：指应用环境中涉及他人证书的用户（user），或证书持有者 subject。

② CA：Certification Authority 的缩写，负责签发证书和 CRL，并将证书和 CRL 发布到存储库中。CA 支持多级，上级 CA 可以签发下级 CA，最下级 CA 只能签发终端实体证书，最顶级 CA 称作根 CA。为实现互联互通，不同 CA 之间可以签发交叉认证证书。

③ RA：Registration Authority 的缩写，属于可选系统。CA 可将部分管理功能授权给 RA 负责，如注册申请、证书发布等。

④ CRL 签发者（CRL Issuer）：属于可选系统。CA 可将 CRL 签发和 CRL 发布功能授权给 CRL 签发者负责。CRL 是 Certificate Revocation List 的缩写，表示证书作废列表。

⑤ 存储库（Repository）：分布式系统的统称，专门用于存储证书和 CRL。终端实体可以通过访问存储库而方便地获得所需的证书和 CRL。由于数字证书具有防伪性和有效期，因此数字证书可以通过任意非信任的系统进行分发，也可以存储到任何非安全的设备中。

2. 证书作废（Revocation）

数字证书签发后，在其有效期内应该处于有效状态。但是，有很多原因可能会导致处

于有效期内的证书不能继续使用，如名称发生改变、CA 和证书持有者 subject 之间的关系发生变化（如员工可能离职）、私钥泄露或被怀疑泄露。如果发生这些情况，CA 需要将该证书作废，使该证书处于无效状态。

CA 引入 CRL 机制用于对外发布作废证书。CRL 是 Certificate Revocation List 的缩写，表示证书作废列表。CRL 跟数字证书类似，是一个特殊的数据结构，由 CA 或 CRL 签发者签名、所有作废证书的序列号等组成，可以以文件形式存在。当应用系统接收到对方证书时，为验证该证书是否有效，不仅需要核对该证书中包含的 CA 签名和有效期，而且需要核对该证书是否已经作废（获取最新 CRL，检查该证书序列号是否在该 CRL 中）。CA 或 CRL 签发者定期签发并发布 CRL，并在 CRL 中指明下次签发的最晚时间。

由于 CRL 也具有防伪性，因此 CRL 也可以跟数字证书类似，通过任意非信任的系统进行分发。

3. 操作协议（Operational Protocols）

操作协议用于将证书和 CRL（或状态信息）传递给客户端系统。可采用多种方式进行传递，如 LDAP、HTTP、FTP、X.500 等。

4. 管理协议（Management Protocols）

管理协议用于 PKI 用户和管理实体之间进行功能交互。管理协议主要包括如下功能集合。

- ① 注册（Registration）：用户将用于申请证书的相关信息提交给 CA。可直接提交，也可通过 RA 间接提交。
- ② 初始化（Initialization）：为保证客户端系统运行时的安全性，在使用前需要预先正确地安装与密钥相关数据。例如，受信任 CA 的相关信息，以及用户自己的公/私钥对。
- ③ 签发证书（Certification）：CA 为用户公钥签发数字证书，并将证书返回给用户的客户端系统，同时将证书发布到存储库。
- ④ 密钥对恢复（Key Pair Recovery）：如果有需要，CA 或密钥备份系统可以备份用户的密钥资料（如私钥）。必要时，可为用户恢复密钥资料，如私钥、口令等。
- ⑤ 密钥对更新（Key Pair Update）：所有的密钥对都需要定期更新成新密钥对，然后签发新证书。
- ⑥ 作废请求（Revocation Request）：当证书出现异常时，用户可向 CA 提出申请，将该证书作废。
- ⑦ 交叉认证（Cross-Certification）：两个 CA 通过互相签发交叉认证证书实现互联互通。

14.2 国内标准

《证书认证系统密码及其相关安全技术规范》定义了 CA 与 KMC 的系统结构。该规范中将 CA 称作证书认证系统，KMC 称作密钥管理系统。KMC 只用于实现双证书机制（签名证书和加密证书）。

14.2.1 证书认证系统 CA

证书认证系统是对生命周期内的数字证书进行全过程管理的安全系统。证书认证系统

- ① 核心层：由证书/CRL 签发系统、证书/CRL 存储发布系统构成。
- ② 管理层：由证书管理系统、安全管理系统构成。
- ③ 服务层：由用户注册管理系统（包括远程用户注册管理系统）、证书/CRL 查询系统构成。其中用户注册管理系统等价于 RA。

证书认证系统的结构如图 14-2 所示。

图 14-2 证书认证系统结构

1. 用户注册管理系统

用户注册管理系统负责用户的证书申请、身份审核和证书下载,可分为本地注册管理系统和远程注册管理系统。

(1) 证书申请

证书申请可采用在线或离线两种方式。在线方式是指,用户通过互联网等登录到用户注册管理系统申请证书;离线方式是指,用户到指定的注册机构申请证书。

(2) 身份审核

审核人员通过用户注册管理系统,对证书申请者进行身份审核。

(3) 证书下载

证书下载可采用在线或离线两种方式。在线方式是指,用户通过互联网等登录到用户注册管理系统下载证书;离线方式是指,用户到指定的注册机构下载证书。

2. 证书/CRL 签发系统

证书/CRL 签发系统负责生成、签发数字证书和 CRL。

(1) 证书类型

按主体对象, 证书通常可分为人员证书、设备证书和机构证书 3 种类型。按功能, 证

书可分为加密证书和签名证书两种类型。具体证书分类方式可根据实际需求进行设计。

(2) 证书机制

当采用双证书机制时,每个用户拥有两张数字证书:一张用于数字签名,另一张用于信息加密。用于数字签名的密钥对可以由用户利用具有密码运算功能的证书载体产生;用于信息加密的密钥对由密钥管理系统产生。签名证书和加密证书一起保存在用户的证书载体中。

(3) 证书签发

用户的数字证书由该系统的CA签发,根CA的数字证书由根CA自己签发,下级CA的数字证书由上级CA签发。

(4) CRL

CRL是在证书有效期之内,CA签发的终止使用证书的信息,分为用户证书作废列表(CRL)和CA证书作废列表(ARL)两类。在证书的使用过程中,应用系统通过检查CRL/ARL,获取有关证书的状态。

3. 证书/CRL存储发布系统

证书/CRL存储发布系统负责数字证书、CRL的存储和发布。

根据应用环境的不同,证书/CRL存储发布系统可采用数据库或目录服务方式,实现数字证书/CRL的存储、备份和恢复等功能,并提供查询服务。

使用目录服务方式,可采用主、从目录结构以保证主目录服务器的安全。同时从目录服务器可以采用分布式的方式进行设置,以提高系统的效率。用户只能访问从目录服务器。

4. 证书/CRL查询系统

证书/CRL查询系统负责为用户和应用系统提供证书状态查询服务,包括:

① CRL查询:用户或应用系统利用数字证书中标识的CRL地址,下载CRL,并检验证书的有效性。

② 在线证书状态查询:用户或应用系统按照OCSP协议,实时在线查询证书的状态。在实际应用中,可以根据具体情况采用上述两种查询方式之一或全部。

5. 证书管理系统

证书管理系统是证书认证系统中实现对证书/CRL的申请、审核、生成、签发、存储、发布、作废、归档等功能的管理控制系统。

6. 安全管理系统

安全管理系统主要包括安全审计系统和安全防护系统。

安全审计系统提供事件级审计功能,对涉及系统安全的行为、人员、时间等记录进行跟踪、统计和分析。

安全防护系统提供访问控制、入侵检测、漏洞扫描、病毒防治等网络安全功能。

14.2.2 密钥管理系统 KMC

密钥管理系统提供了对生命周期内的加密证书密钥对进行全过程管理的功能,包括密钥生成、密钥存储、密钥分发、密钥备份、密钥更新、密钥撤销、密钥归档、密钥恢复以

及安全管理等。

密钥管理系统的结构如图 14-3 所示。

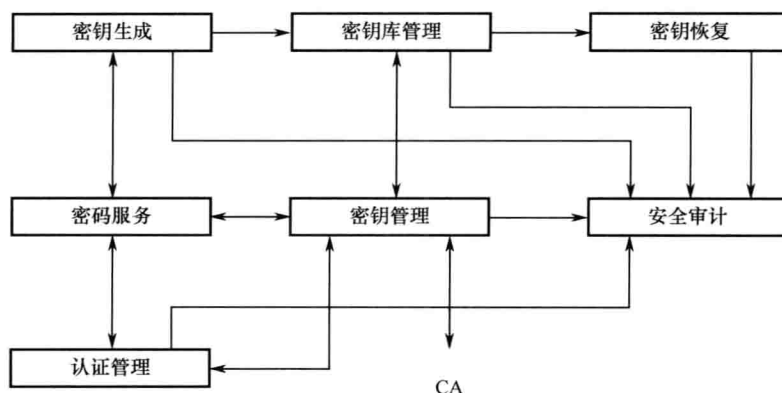


图 14-3 密钥管理系统结构

1. 密钥生成

根据 CA 的请求为用户生成非对称密钥对（公私钥对），该密钥对应该由密钥管理系统的硬件密码设备生成。

2. 密钥存储

密钥管理系统生成的非对称密钥对，经硬件密码设备加密后存储在数据库中。

3. 密钥分发

密钥管理系统生成的非对称密钥对通过证书认证系统分发到用户证书载体中。

4. 密钥备份

密钥管理系统采用热备份、冷备份和异地备份等措施实现密钥备份。

5. 密钥更新

当证书到期或用户需要时，密钥管理系统根据 CA 请求为用户生成新的非对称密钥对。

6. 密钥撤销

当证书到期、用户需要或管理机构认为必要时，密钥管理系统根据 CA 请求撤销用户当前使用的密钥。

7. 密钥归档

密钥管理系统为到期或撤销的密钥提供安全的长期存储。

8. 密钥恢复

密钥管理系统可为用户提供密钥恢复服务和为司法取证提供密钥恢复服务。密钥恢复需按管理策略进行审批，一般用户只限于恢复自身密钥。

第15章 系统设计

系统设计包括系统的整体设计和各子系统设计,《证书认证系统密码及其相关安全技术规范》定义了 CA 与 KMC 系统的设计原则和各子系统的实现方式。在具体实现过程中,应根据所选择的开发平台和开发环境进行详细设计。

15.1 证书认证系统 CA

证书认证系统应遵循以下总体设计原则:

- ① 证书认证系统遵循标准化、模块化设计原则。
- ② 证书认证系统设置相对独立的功能模块,通过各模块之间的安全连接,实现各项功能。
- ③ 各模块之间的通信采用基于身份验证机制的安全通信协议。
- ④ 各模块使用的密码运算都应该在密码设备中完成。
- ⑤ 各模块产生的审计日志文件采用统一的格式传递和存储。
- ⑥ 用户注册管理系统、证书/CRL 签发系统和密钥管理系统可以设置独立的数据库。
- ⑦ 系统必须具备访问控制功能。
- ⑧ 系统在实现证书管理功能的同时,必须充分考虑系统本身的安全性。

15.1.1 用户注册管理系统 RA

1. 系统功能

用户注册管理系统负责用户证书/CRL 的申请、审核以及证书的制作,其主要功能如下。

(1) 用户信息的录入

录入用户的申请信息。用户申请信息包括签发证书所需要的信息,还包括用于验证用户身份的信息,这些信息存放在用户注册管理系统的数据库中。用户注册管理系统应能够批量接受从外部系统生成的、以电子文档方式存储的用户信息。

(2) 用户信息的审核

提取用户的申请信息,审核用户的真实身份,当审核通过后,将证书签发所需要的信息提交给签发系统。

(3) 用户证书下载

用户注册管理系统提供证书下载功能,当签发系统为用户签发证书后,用户注册管理系统能够下载用户证书,并将用户证书写入指定的用户证书载体中,然后分发给用户。

(4) 安全审计

负责对用户注册管理系统的管理人员、操作人员的操作日志进行查询、统计以及报表打印等。

(5) 安全管理

对用户注册管理系统的登录进行安全访问控制,并对用户信息数据库进行管理和备份。

(6) 多级审核

用户注册管理系统可根据需要采用分级部署的模式，对不同种类和等级的证书，可由不同级别的用户注册管理系统进行审核。用户注册管理系统应能够根据需求支持多级注册管理系统的建立和多级审核模式。

用户注册管理系统应具有并行处理的能力。

2. 模块组成

用户注册管理系统由本地注册管理、远程注册管理、数据库、信息录入、身份审核、证书制作、安全管理及安全审计几部分构成，其逻辑结构如图 15-1 所示。

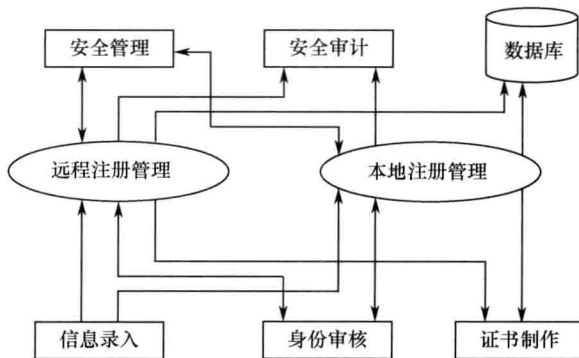


图 15-1 用户注册管理系统的逻辑结构

15.1.2 证书/CRL 签发系统

15.1.2.1 系统功能

证书/CRL 签发系统是证书认证系统的核心，不仅为整个证书认证系统提供签发证书/CRL 的服务，还承担整个证书认证系统中主要的安全管理工作。其主要功能如下。

(1) 证书生成与签发

从数据库中读取用户信息，根据拟签发的证书类型向密钥管理系统申请加密密钥对，生成用户的签名证书和加密证书，将签发完成的证书发布到目录服务器和数据库中。根据系统的配置和管理策略，不同种类或用途的证书可以采用不同的签名密钥。

(2) 证书更新

系统应提供 CA 证书及用户证书的更新功能。

(3) CRL 生成与签发

接收作废信息，验证作废信息中的签名，然后签发 CRL，将签发后的 CRL 发布到数据库或目录服务器中。签发 CRL 的签名密钥可以与签发证书的签名密钥相同或不同。

(4) 安全审计

负责对证书/CRL 生成与签发系统的管理人员、操作人员的操作日志进行查询、统计以及报表打印等。

(5) 安全管理

对证书/CRL 生成与签发系统的登录进行安全访问控制，并对证书/CRL 数据库进行管理和备份；设置管理员、操作员，并为这些人员申请和下载数字证书；配置不同的密码设

备：配置不同的证书模板。

证书/CRL 生成与签发系统应具有并行处理的能力。

15.1.2.2 模块组成

证书/CRL 签发系统由证书/CRL 生成与签发、安全管理、安全审计、数据库、目录服务器以及密码设备等组成。

1. 证书/CRL 生成与签发

主要功能包括证书的生成与签发和 CRL/ARL 的生成与签发。

(1) 证书的签发

根据接收的请求信息，从数据库中提取用户的信息，向密钥管理系统申请加密密钥对，然后生成并签发签名证书和加密证书，签发的证书和加密证书的私钥通过证书管理系统下传给申请者，同时将证书发布到数据库和目录服务器中。在此过程中，必须保证私钥传递的安全。

(2) CRL 的签发

首先验证申请信息中的数字签名，然后签发 CRL，并将 CRL 发布到数据库或目录服务器指定的位置。

2. 密码设备

密码设备完成签名以及验证工作，并负责与其他系统通信过程中的密码运算，CA 的签名密钥保存在密码设备中。在进行上述工作时，必须保证所使用的密钥不能以明文形式被读出密码设备。

3. 安全管理

安全管理主要包括以下内容。

① 证书模板配置：不同的证书种类由不同的证书模板确定，证书模板包括相应种类证书的基本项和证书的扩展项。

② CRL 发布策略配置：配置 CRL 的发布策略，包括自动/人工发布模式选择、发布时间间隔。

③ 进行 CA 密钥的更新。

④ 进行证书的备份和归档。

⑤ 进行服务器安全配置，包括服务器可接受的主机访问列表。

⑥ 为其他子系统定义管理员以及为这些管理员签发数字证书。

⑦ 数据库系统的配置：数据源的选择，数据库连接的用户名和口令设置。

4. 安全审计

查询证书/CRL 生成与签发系统中的安全审计日志，并进行统计与打印。

15.1.3 证书/CRL 存储发布系统

1. 系统功能

证书/CRL 存储发布系统负责证书和 CRL 的存储与发布，是证书认证系统的基础组成部分。证书的存储和发布必须采用数据库、目录服务器或其中之一。

该系统主要功能如下：

- ① 证书存储。
- ② CRL 存储。
- ③ 证书和 CRL 发布。
- ④ 安全审计：负责对证书/CRL 存储发布系统的管理人员、操作人员的操作日志进行查询、统计及报表打印等。
- ⑤ 安全管理：对证书/CRL 存储发布系统的登录进行访问控制，并定期对数据库和目录服务器进行管理和备份。
- ⑥ 数据一致性检验：对数据库和目录服务器中的数据进行一致性检验。

2. 模块组成

证书/CRL 存储发布系统由数据库或主/从目录服务器、安全管理模块、安全审计模块组成。

(1) 数据库

存放证书和 CRL 及用户的其他信息。

(2) 目录服务器

证书/CRL 存储发布系统采用主从结构的目录服务器，签发完成的数据直接写入主目录服务器中，然后由目录服务器的主从映射功能自动映射到从目录服务器中。主从目录服务器通常配置在不同等级的安全区域，用户只能访问从目录服务器。

(3) 安全管理

安全管理主要包括以下内容。

- ① 定期对数据库和目录服务器的内容进行数据备份和归档。
- ② 对数据库和目录服务器中的数据进行一致性检查，发现不一致时，应进行数据恢复。

(4) 安全审计

查询证书/CRL 存储与发布系统中的安全审计日志，并进行统计与打印等。

15.1.4 证书/CRL 查询系统

1. 系统功能

证书/CRL 查询系统为用户及应用系统提供证书状态查询服务，可以采用以下两种方式。

(1) CRL 查询

用户或应用系统利用证书中标识的 CRL 地址，查询并下载 CRL 到本地，进行证书状态的检验。

(2) 在线证书状态查询

用户或应用系统利用 OCSP 协议，在线实时查询证书的状态，查询结果经过签名后返回给请求者，进行证书状态的检验。

2. 模块组成

证书/CRL 查询系统由证书状态数据库/OCSP 服务器、安全管理模块、安全审计模块以及密码设备组成。

(1) 证书状态数据库/OCSP 服务器

接受用户及应用系统的证书状态查询请求，根据请求信息中的证书序列号，从证书状

态数据库中查询证书的状态，并将查询结果返回给请求者。

(2) 密码设备

验证请求信息中的签名，并对查询结果进行签名。

(3) 安全管理

安全管理主要包括以下内容。

① OCSP 服务器的配置，定义可接受的访问控制信息以及查询的证书状态数据库的地址。

② 启动/停止查询服务，配置可接受的用户请求数量等。

(4) 安全审计

查询证书状态查询系统中的安全审计日志，并进行统计与打印等。

15.1.5 证书管理系统

证书管理系统是证书认证系统的综合信息控制和调度服务系统，它接收用户的各种请求信息，并将请求信息提交给相应的子系统。

证书管理系统是一个逻辑上独立的系统，在进行系统设计过程中，可根据证书认证系统提供的服务，由不同的处理模块组成。这些模块可以采用分布式结构，以增强系统的处理能力，提高系统的效率。

15.1.6 安全管理系统

安全管理系统主要包括安全审计系统和安全防护系统。

1. 安全审计系统

提供事件级审计功能，对涉及系统安全的行为、人员、时间的记录进行跟踪、统计和分析。安全审计系统可以分别查询各子系统日志记录，也可以通过查询证书/CRL 存储与发布系统中的数据库进行集中审计。

日志记录的主要内容包括：

- ① 操作员姓名。
- ② 操作项目。
- ③ 操作起始时间。
- ④ 操作终止时间。
- ⑤ 证书序列号。
- ⑥ 操作结果。

日志管理的主要内容包括：

- ① 日志参数设置。设置日志保存的最大规模和日志备份的目录。
- ② 日志查询。查询操作员、操作事件信息。
- ③ 日志备份。当日志保存到日志参数设置的最大规模时，将保存的日志备份。
- ④ 日志处理。对日志记录的正常业务流量和各类事件进行分类整理。
- ⑤ 证据管理。对证据数据进行审计、统计和记录。

2. 安全防护系统

提供访问控制、入侵检测、漏洞扫描、病毒防治等网络安全功能。

15.2 密钥管理系统 KMC

密钥管理系统应遵循以下总体设计原则：

- ① 密钥管理系统遵循标准化、模块化设计原则。
- ② 密钥管理系统设置相对独立的功能模块，通过各模块之间的安全连接，实现各项功能。
- ③ 各模块之间的通信采用基于身份验证机制的安全通信协议。
- ④ 各模块使用的密码运算都必须在密码设备中完成。
- ⑤ 各模块产生的审计日志文件采用统一的格式传递和存储。
- ⑥ 系统必须具备访问控制功能。
- ⑦ 系统在实现密钥管理功能的同时，必须充分考虑系统本身的安全性。
- ⑧ 系统可为多个 CA 提供密钥服务。当为多个 CA 提供密钥服务时，由上级 CA 为密钥管理系统签发证书。

1. 密钥生成模块

密钥生成模块应提供以下主要功能：

- ① 非对称密钥对的生成，并将其保存在备用库中。当备用库中密钥数量不足时，自动进行补充。
- ② 对称密钥的生成。
- ③ 随机数的生成。

2. 密钥管理模块

密钥管理模块应提供以下主要功能：

- ① 接收、审核 CA 的密钥申请。
- ② 调用备用密钥库中的密钥对。
- ③ 向 CA 发送密钥对。
- ④ 对调用的备用密钥库中的密钥对进行处理，并将其转移到在用密钥库。
- ⑤ 对在用密钥库中的密钥进行定期检查，将超过有效期的或被撤销的密钥转移到历史密钥库。
- ⑥ 对历史密钥库中的密钥进行处理，将超过规定保留期的密钥转移到规定载体。
- ⑦ 接收与审查关于恢复密钥的申请，依据安全策略进行处理。
- ⑧ 对进入本系统的有关操作及操作人员进行身份与权限的认证。

3. 密钥库管理模块

密钥库管理模块负责密钥的存储管理，按照其存储的密钥的状态，密钥库分为备用库、在用库和历史库 3 种类型，密钥库中的密钥数据必须加密存放。

(1) 备用库

备用库存放待使用的密钥对。密钥生成模块预生成一批密钥对，存放于备用库中；CA

需要时,可及时调出,将其提供给CA后转入在用库。

备用密钥库应保持一定数量的待用密钥对,存放的密钥数量依系统的用户数量而定,若少于设定的最低数量则应自动补足到规定数量。

(2) 在用库

在用库存放当前使用的密钥对。在用库中的密钥记录包含用户证书的序列号、ID号和有效时间等标志。

(3) 历史库

历史库存放过期或已被撤销的密钥对。历史库中的密钥记录包含用户证书的序列号、ID号有效时间和作废时间等标志。

4. 认证管理模块

认证管理模块负责对进入本系统的有关操作及操作人员进行身份与权限的认证。

5. 安全审计模块

安全审计模块负责各个功能模块的运行事件检查、有关资料分析和密钥申请统计等服务。审计项目主要包括:

- ① 运行事件记录。
- ② 服务器状态记录。
- ③ 系统重要策略设置。

审计记录不能进行修改。

6. 密钥恢复模块

密钥恢复模块负责为用户和司法取证恢复用户的加密私钥,被恢复的私钥必须安全地下载到载体。

(1) 用户密钥恢复

用户通过RA申请,经审核后,由CA向密钥管理系统提出密钥恢复请求,密钥恢复模块恢复用户的密钥并通过CA返回RA,下载于用户证书载体中。

(2) 司法取证密钥恢复

司法取证人员必须到KMC进行司法取证密钥恢复,KMC对司法取证人员的身份进行认证,认证通过后,由密钥恢复模块恢复所需的密钥并下载于特定载体中。

7. 密码服务模块

密码服务模块负责为密钥管理系统的各项业务提供密码支持。

密码服务模块配置经国家密码主管部门审批的非对称密钥密码算法、对称密钥密码算法和数据摘要算法等。

密码算法必须在硬件密码设备中运行。

8. 审计模块

密钥管理系统设置日志审计模块,包括全程审计和事件审计。审计员定时调出审计记录,制作统计分析表。审计员可以处理但不能修改日志审计数据。

日志记录的主要内容包括:

- ① 操作员姓名。

- ② 操作项目。
- ③ 操作起始时间。
- ④ 操作终止时间。
- ⑤ 证书序列号。
- ⑥ 操作结果。

日志管理的主要内容包括：

- ① 日志参数设置。设置日志保存的最大规模和日志备份的目录。
- ② 日志查询。日志查询主要是查询操作员、认证机构操作事件信息。
- ③ 日志备份。当保存的日志达到参数设置的最大规模时，将对已有日志进行备份。
- ④ 日志处理。对日志记录的正常业务流量和各类事件进行分类整理。
- ⑤ 证据管理。对证据数据进行审计、统计和记录。

15.3 企业级 CA 总体设计示例

从管理主体和服务对象分析，一般可将 CA 系统分为两类：运营型 CA 和企业级 CA。

运营型 CA 通常以第三方电子认证服务机构（CA 中心）方式存在，并由其运营和管理，提供具有法律效力的第三方电子认证服务，并承担相应的法律责任；不仅需要构建专业的 CA 机房和健全的法律文件，而且还需要确保 CA 系统和认证业务的安全性，同时还要满足物理环境与设施、组织与人员管理、文档/记录与介质管理、业务连续性、审计与改进、认证服务性能等方面的多种要求。运营型 CA 可以为多个行业的多种应用提供服务，具有很强的通用性和独立性；证书规模至少在几十万以上，投资规模也比较庞大。

企业级 CA 通常由企业内部特定部门或团队维护和管理，只是作为一类安全系统对待，只为该企业自身的 IT 系统提供应用安全服务，证书规模一般在几万以内，投资规模较小。企业级 CA 系统不仅需要同企业的各种应用进行深度融合，而且还需要适应组织内部的行政管理模式，在灵活部署、方便使用、易于扩展等方面要求较高。

本节主要介绍企业级 CA 系统总体设计的一种示例和实践，仅供参考。

15.3.1 技术路线选择

1. 通用性技术路线

- ① 管理 UI：采用 B/S 模式。
- ② 运营平台：操作系统优先使用 Linux，应支持多种操作系统；数据库优先使用 MySQL，应支持多种数据库，通过 JDBC 访问数据库；应用中间件优先使用 Tomcat。
- ③ 开发工具：前台模块优先使用 C、C++或 VC，如控件、客户端工具等；后台 Web 服务模块优先使用 JSP、Servlet 或 JDK。

2. 专业性技术路线

（1）证书机制

- ① 证书分类：分为个人、单位、Web、设备等，可扩展。
- ② 证书状态：包括正常、作废、冻结、解冻、过期等。

③ 证书业务状态：分为证书申请、证书作废、证书冻结、证书解冻、证书更新等。其中，证书申请状态包括已录入、审核通过、审核失败、已制作；证书作废状态包括已录入、审核通过、审核失败；证书冻结状态包括已录入、审核通过、审核失败；证书解冻状态包括已录入、审核通过、审核失败；证书更新状态包括已录入、审核通过、审核失败、已更新。

④ 证书申请流程：分为三步流程和一步流程两种。三步流程由录入、审核、制作组成，每步流程分别由不同业务员进行操作；一步流程由同一个业务员一次性完成。

⑤ 证书更新流程：分为三步流程、二步流程和一步流程3种。三步流程由录入、审核、更新组成，二步流程由审核（基于旧证书）、更新组成，每步流程分别由不同业务员进行操作；一步流程由同一个业务员一次性完成。证书制作时可通过旧证书对证书申请者进行身份认证。

⑥ 证书作废/冻结/解冻流程：分为二步流程和一步流程两种。二步流程由录入、审核、组成，每步流程分别由不同业务员进行操作；一步流程由同一个业务员一次性完成。

(2) 管理机制

① 三级管理：分为CA管理员、RA管理员、RA操作员3级。

② 身份认证：支持证书模式和口令模式。

③ 访问控制：采用RBAC机制。

④ CA管理员：主要职责是对CA进行管理，包括初始化、CA策略管理、RA管理、RA管理员管理、查询统计、强制业务功能等。

⑤ RA管理员：主要职责是对RA操作员进行管理。

⑥ RA操作员：主要职责是进行具体业务操作，又分为3种角色：录入员、审核员、制作员。

具体管理机制如图15-2所示。

(3) RA服务

① 业务功能：主要包括证书申请、证书作废、证书更新、证书冻结、RA操作员管理、查询统计、RA策略管理等。

② RA管理员：主要职责包括操作员管理、查询统计、RA策略配置、强制业务功能等。

③ RA录入员：主要职责包括查询统计、录入等。

④ RA审核员：主要职责包括查询统计、审核等。

⑤ RA制作员：主要职责包括查询统计、制作等。

(4) 证书服务

分为Web服务类和非Web服务类。

① Web服务类：主要包括CA证书下载（支持证书链）、CRL下载、单证书下载/更新、双证书下载/更新、Web证书下载、证书查询、证书申请/更新自助录入、证书作废/冻结/解冻自助录入等。

② 非Web服务类：主要包括OCSP/SOCSP服务、LDAP证书服务、证书验证服务等。

(5) 安全性

① License控制：通过证书数和RA数控制License。

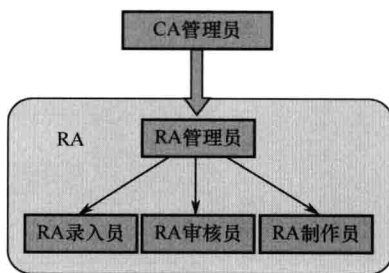


图 15-2 企业级 CA 管理机制

② Web 方式管理：身份认证优先采用证书认证方式（双向 SSL 或单向 SSL+安全控件），也可以支持口令方式。

③ 用户自助下载证书：采用口令方式进行身份认证。

④ 口令存储安全性：数据库中应该加密存储。

⑤ 关键数据存储完整性：采用 HMAC 机制。

⑥ 关键数据传输机密性：在 RA 与 CA 间采用对称加密机制。

15.3.2 模块设计

企业级 CA 系统模块组成如图 15-3 所示，其中以 w 开头的模块表示以 Web 方式提供服务。

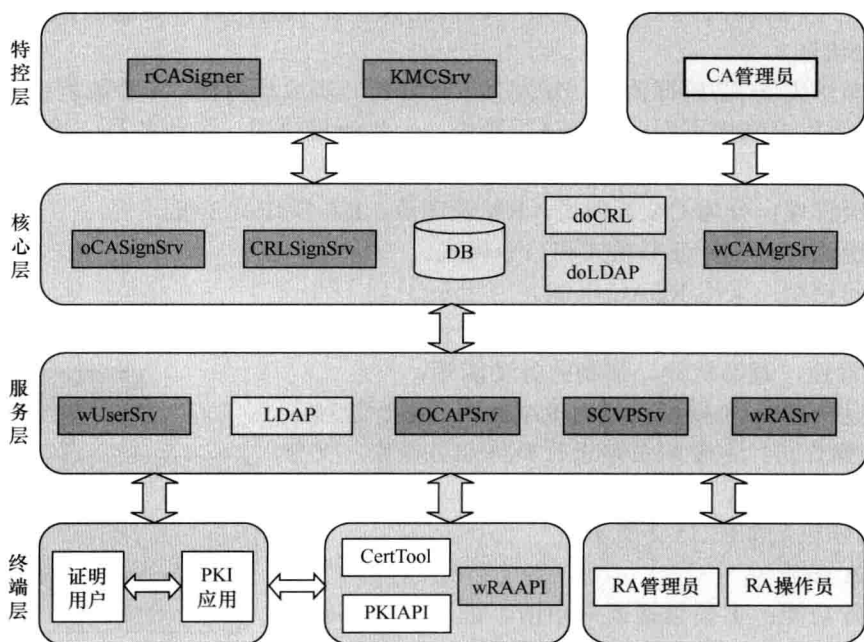


图 15-3 企业级 CA 系统模块组成

各模块主要功能如下。

① rCAsigner：主要功能包括产生根 CA 公私钥对、签发根 CA 证书、签发运营 CA 证书、签发 CRL 签名者证书、支持软件 HSM 和硬件 HSM 等。

② KMCSrv：主要功能包括对外提供加密公私钥对产生服务、支持软件 HSM 和硬件 HSM 等。

③ oCAsignSrv：主要功能包括产生运营 CA 公私钥对、对外提供证书签发服务、支持单证书和双证书签发、支持软件 HSM 和硬件 HSM、对外提供 License 验证服务等。

④ CRLSignSrv：主要功能包括产生 CRL 签名者公私钥对、对外提供 CRL 签发服务等。

⑤ wCAMgrSrv：主要功能包括配置 CA 参数、管理 RA、管理 CA 管理员和 RA 管理员、证书管理高级功能（强制申请、强制作废、强制冻结、强制解冻、强制审核、强制更新等）等。

⑥ doCRL：主要功能包括定时从数据库提取证书，并提交 CRLSignSrv 签名后，保存到数据库或文件中。

⑦ doLDAP: 主要功能包括定时从数据库提取证书, 发布到 LDAP 中。

⑧ DB: 优先使用 MySQL。

⑨ wUserSrv: 主要功能包括下载 CA 证书 (根 CA、运营 CA)、下载 CRL、根据条件查询并下载他人证书、自身证书申请 (产生公私钥对、提交 P10 包、安装证书等)、自身证书更新 (通过旧证书进行身份认证) 等。

⑩ LDAP: 优先使用 OpenLDAP。

⑪ OCSPSrv: 基于 OCSP (Online Certificate Status Protocol) 协议提供在线证书状态查询服务。

⑫ SCVPSrv: 基于 SCVP (Server-based Certificate Validation Protocol) 协议提供在线证书验证服务。

⑬ wRASrv: 主要功能包括管理 RA 操作员、证书管理普通功能 (证书申请、证书作废、证书冻结、证书解冻、证书更新等)、证书管理高级功能 (强制申请、强制作废、强制冻结、强制解冻、强制审核、强制更新等) 等。

⑭ wRAAPI: 对外提供 API 方式, 方便将 RA 功能与应用系统集成, 主要包括强制申请、强制作废、强制冻结、强制解冻、强制审核、强制更新等证书管理高级功能。

⑮ CertTool: 主要功能包括查看证书信息、产生或拆分 P12 证书链、各种加密算法测试等。

各模块开发工具、依托资源和存储方式如表 15-1 所示。

表 15-1 企业级 CA 系统开发工具、依托资源和存储方式

/	模块名称	开发工具	依托资源	存储方式
1	rCASigner	Ansi C	openssl	文件系统
2	KMCSrv	Ansi C	openssl	文件系统
3	oCASignSrv	Ansi C	openssl	文件系统
4	CRLSignSrv	Ansi C	openssl	文件系统
5	wCAMgrSrv	Java	Tomcat+JRE+JDBC+MySQL	数据库
6	doCRL	Java	Tomcat+JRE+JDBC+MySQL	数据库
7	doLDAP	Java	Tomcat+JRE+JDBC+MySQL	数据库
8	DB	/	MySQL	数据库
9	wUserSrv	Java	Tomcat+JRE+JDBC+MySQL	数据库
10	LDAP	Java	LDAP	数据库
11	OCSPSrv	Java	Tomcat+JRE+JDBC+MySQL	数据库
12	SCVPSrv	Java	Tomcat+JRE+JDBC+MySQL	数据库
13	wRASrv	Java	Tomcat+JRE+JDBC+MySQL	数据库
14	wRAAPI	Java、Ansi C	/	待定
15	CertTool	VC++	openssl	/

15.3.3 数据库设计

数据库表设计如表 15-2 所示。

表 15-2 企业级 CA 数据库表

/	表名称	主要目的	/	表名称	主要目的
1	caconf	保存系统各种配置参数	5	userinfo	保存用户基本信息
2	rainfo	保存 RA 信息	6	userbiz	保存用户证书管理信息
3	opinfo	保存操作员信息	7	usercert	保存用户证书信息
4	loginfo	保存操作日志	8	cacrl	保存 CRL 信息

假设数据库名称、密码和用户暂时设置为 appca、appca 和 caadmin（可以修改）。数据库表结构设计如下：

```
drop database appca;
drop user caadmin;
create database appca;
use appca;
CREATE TABLE `caconf` (`confid` int NOT NULL, `confvalue` text NOT NULL, `confstatus`
int NOT NULL, `remark` text DEFAULT NULL, PRIMARY KEY (`confid`)) ENGINE=InnoDB DEFAULT
CHARSET=utf8;
```

```
CREATE TABLE `rainfo` (`raid` varchar(100) NOT NULL, `raename` varchar(100) NOT NULL,
`racontactor` varchar(100) DEFAULT NULL, `ratel` varchar(100) DEFAULT NULL, `ramp` varchar(100)
DEFAULT NULL, `raemail` varchar(100) DEFAULT NULL, `rastatus` int NOT NULL, `ranotbefore`
varchar(25) DEFAULT NULL, `ranotafter` varchar(25) DEFAULT NULL, `macvalue` varchar(100)
DEFAULT NULL, `fromtime` varchar(25) NOT NULL, `lasttime` varchar(100) NOT NULL, `lastopid`
varchar(100) NOT NULL, `remark` text DEFAULT NULL, PRIMARY KEY (`raid`)) ENGINE=InnoDB
DEFAULT CHARSET=utf8;
```

```
CREATE TABLE `opinfo` (`opid` varchar(100) NOT NULL, `op2raid` varchar(100) NOT NULL,
`opname` varchar(100) NOT NULL, `oporg` varchar(100) DEFAULT NULL, `oporgunit` varchar(100)
DEFAULT NULL, `oporgpos` varchar(100) DEFAULT NULL, `optel` varchar(100) DEFAULT NULL, `opmp`
varchar(100) DEFAULT NULL, `opemail` varchar(100) DEFAULT NULL, `opstatus` int NOT NULL, `optype`
int NOT NULL, `opauthtype` int NOT NULL, `opnotbefore` varchar(25) DEFAULT NULL, `opnotafter`
varchar(25) DEFAULT NULL, `opcerts` varchar(100) DEFAULT NULL, `opcertsnotbefore` varchar(25)
DEFAULT NULL, `opcertsnotafter` varchar(25) DEFAULT NULL, `opcervalue` text DEFAULT NULL,
`opcerthash` varchar(100) DEFAULT NULL, `opcode` varchar(100) NOT NULL, `oppin` varchar(100) NOT
NULL, `macvalue` varchar(100) DEFAULT NULL, `fromtime` varchar(25) NOT NULL, `lasttime` varchar(25)
NOT NULL, `lastopid` varchar(100) NOT NULL, `remark` text DEFAULT NULL, PRIMARY KEY (`opid`))
ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
CREATE TABLE `loginfo` (`logid` varchar(100) NOT NULL, `log2raid` varchar(100) NOT NULL,
`logtype` int NOT NULL, `logresult` int NOT NULL, `logresultinfo` text DEFAULT NULL, `logsubjecttype`
int NOT NULL, `logsubjectid` varchar(100) NOT NULL, `logsubjectinfo` text DEFAULT NULL, `logobjectid`
varchar(100) DEFAULT NULL, `logobjectinfo` text DEFAULT NULL, `logtime` varchar(25) NOT NULL,
`logopcervalue` text DEFAULT NULL, `logopcerts` varchar(100) DEFAULT NULL, `logopcerts` varchar(100)
DEFAULT NULL, PRIMARY KEY (`logid`)) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
CREATE TABLE `userinfo` (`userid` varchar(100) NOT NULL, `user2raid` varchar(100) NOT NULL,
```

```
`usertype` int NOT NULL, `userstatus` int NOT NULL, `usercn` varchar(100) NOT NULL, `usero` varchar(100)
DEFAULT NULL, `userou` varchar(100) DEFAULT NULL, `userorgpos` varchar(100) DEFAULT NULL,
`usertel` varchar(100) DEFAULT NULL, `usermp` varchar(100) DEFAULT NULL, `usere` varchar(100)
DEFAULT NULL, `userfax` varchar(100) DEFAULT NULL, `userc` varchar(100) DEFAULT NULL, `userst`
varchar(100) DEFAULT NULL, `userl` varchar(100) DEFAULT NULL, `userw` varchar(100) DEFAULT NULL,
`usercode` varchar(100) NOT NULL, `userpin` varchar(100) NOT NULL, `macvalue` varchar(100) DEFAULT
NULL, `fromtime` varchar(25) NOT NULL, `lasttime` varchar(25) NOT NULL, `lastopid` varchar(100) NOT
NULL, `remark` text DEFAULT NULL, PRIMARY KEY (`userid`) ) ENGINE=InnoDB DEFAULT
CHARSET=utf8;
```

```
CREATE TABLE `userbiz` ( `bizid` varchar(100) NOT NULL, `biz2raid` varchar(100) NOT NULL,
`biz2userid` varchar(100) NOT NULL, `biz2certid` varchar(100) DEFAULT NULL, `biz2certidref` varchar(100)
DEFAULT NULL, `biztype` int NOT NULL, `bizstatus` int NOT NULL, `bizrefcode` varchar(100) DEFAULT
NULL, `bizauthcode` varchar(100) DEFAULT NULL, `bizremark` text DEFAULT NULL, `bizinputopid`
varchar(100) DEFAULT NULL, `bizinputtime` varchar(25) DEFAULT NULL, `bizcheckopid` varchar(100)
DEFAULT NULL, `bizchecktime` varchar(25) DEFAULT NULL, `bizmakeopid` varchar(100) DEFAULT
NULL, `bizmaketime` varchar(25) DEFAULT NULL, `macvalue` varchar(100) DEFAULT NULL, `remark` text
DEFAULT NULL, PRIMARY KEY (`bizid`) ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
CREATE TABLE `usercert` ( `certid` varchar(100) NOT NULL, `cert2raid` varchar(100) NOT NULL,
`cert2userid` varchar(100) NOT NULL, `certtype` int NOT NULL, `certnum` int NOT NULL, `certstatus` int
NOT NULL, `certreqtype` int NOT NULL, `certcrlstatus` int NOT NULL, `certldapstatus` int NOT NULL,
`certcrlcode` int DEFAULT -1, `certcrlremark` text DEFAULT NULL, `certsigntime` varchar(25) DEFAULT
NULL, `certrevoketime` varchar(25) DEFAULT NULL, `certholdtime` varchar(25) DEFAULT NULL, `certc`
varchar(100) NOT NULL, `certst` varchar(100) DEFAULT NULL, `certl` varchar(100) DEFAULT NULL,
`certo` varchar(100) DEFAULT NULL, `certou` varchar(100) DEFAULT NULL, `certe` varchar(100)
DEFAULT NULL, `certcn` varchar(100) NOT NULL, `certw` varchar(100) DEFAULT NULL, `certnotbefore`
varchar(25) DEFAULT NULL, `certnotafter` varchar(25) DEFAULT NULL, `certvaluesign` text DEFAULT
NULL, `certvalueenc` text DEFAULT NULL, `certtokenid` varchar(100) DEFAULT NULL, `certreqp10` text
DEFAULT NULL, `certsnsign` varchar(100) DEFAULT NULL, `certsnenc` varchar(100) DEFAULT NULL,
`macvalue` varchar(100) DEFAULT NULL, `fromtime` varchar(25) NOT NULL, `lasttime` varchar(25) NOT
NULL, PRIMARY KEY (`certid`) ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
CREATE TABLE `cacrl` ( `crlid` int NOT NULL, `crltype` int NOT NULL, `crlvalue` mediumtext
NOT NULL, PRIMARY KEY (`crlid`) ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
insert into opinfo values('20100721appca0000', '11110000', 'camgr', "", "", "", "", "11, 17, 1, '20100721',
", "", "", "", 'camgr', 'camgr', "", '20100721', '20100721', '11110000', "");
```

```
create user caadmin;
```

```
grant all privileges on *.* to caadmin@localhost identified by 'appca' with GRANT OPTION;
```

```
grant all privileges on *.* to caadmin@"%" identified by 'appca' with GRANT OPTION;
```

15.3.4 双证书技术流程设计

双证书技术流程设计如图 15-4 所示。

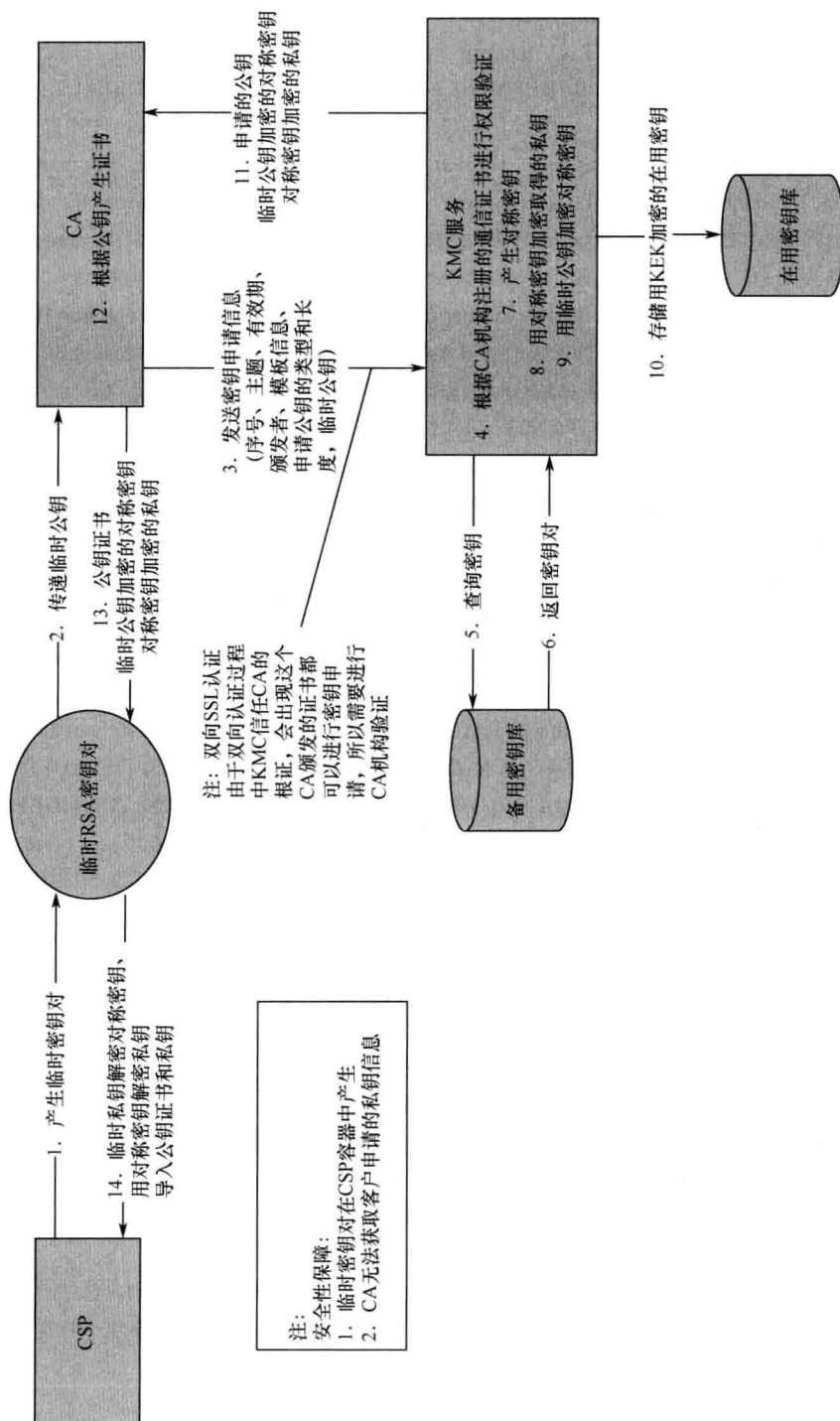


图 15-4 双证书技术流程图

双证书具体技术流程如下：

- ① 用户 CSP 产生临时密钥对。
- ② 用户 CSP 将临时公钥传递给 CA 系统。
- ③ CA 向 KMC 发送密钥申请信息，包括序号、主题、有效期、申请公钥类型和长度、临时公钥等。
- ④ KMC 根据 CA 机构注册的通信证书验证 CA 密钥申请的合法性。
- ⑤ KMC 从备用密钥库查询密钥对。
- ⑥ 备用密钥库将正式密钥对返回 KMC。
- ⑦ KMC 产生对称密钥。
- ⑧ KMC 用对称密钥加密正式私钥。
- ⑨ KMC 用临时公钥加密对称密钥。
- ⑩ KMC 将正式密钥对用 KEK 加密后存储到在用密钥库（KEK 是对密钥进行安全加密的专用密钥，需要预先设置）。
- ⑪ KMC 将正式公钥、对称密钥密文（临时公钥加密）、正式私钥密文（对称密钥加密）等返回给 CA。
- ⑫ CA 根据正式公钥产生用户正式公钥证书。
- ⑬ CA 将正式公钥证书、对称密钥密文（临时公钥加密）、正式私钥密文（对称密钥加密）返回给用户 CSP。
- ⑭ 用户 CSP 使用临时私钥解密获得对称密钥明文，用对称密钥解密获得正式私钥明文，然后将正式公钥证书和正式私钥导入密码介质。

第 16 章 对外在线服务

16.1 OCSP/SOCSP 服务

16.1.1 OCSP

OCSP 是 Online Certificate Status Protocol 的缩写,表示在线证书状态协议。CA 系统通过 OCSP 机制为用户提供在线证书状态查询服务。用户端将待查询证书序列号按照 OCSP 协议组织成 OCSP 请求包,然后将 OCSP 请求包发送给 OCSP 服务器,OCSP 服务器查询数据库获得该序列号对应证书的状态,并组织成 OCSP 响应包后返回给用户端。用户端解析 OCSP 响应包后获得该证书的当前状态。证书状态包括有效、已作废、已冻结等。

IETF RFC 2560 规定了 OCSP 请求包和响应包的具体格式。

1. OCSP 请求包

OCSP 请求包格式用 ASN.1 描述如下:

```
OCSPRequest ::= SEQUENCE {
    tbsRequest          TBSRequest,
    optionalSignature    [0] EXPLICIT Signature OPTIONAL }
TBSRequest ::= SEQUENCE {
    version              [0] EXPLICIT Version DEFAULT v1,
    requestorName        [1] EXPLICIT GeneralName OPTIONAL,
    requestList          SEQUENCE OF Request,
    requestExtensions    [2] EXPLICIT Extensions OPTIONAL }
Signature ::= SEQUENCE {
    signatureAlgorithm    AlgorithmIdentifier,
    signature             BIT STRING,
    certs                [0] EXPLICIT SEQUENCE OF Certificate OPTIONAL }
Version ::= INTEGER { v1(0) }
Request ::= SEQUENCE {
    reqCert              CertID,
    singleRequestExtensions [0] EXPLICIT Extensions OPTIONAL }
CertID ::= SEQUENCE {
    hashAlgorithm         AlgorithmIdentifier,
    issuerNameHash        OCTET STRING, -- Hash of Issuer's DN
    issuerKeyHash         OCTET STRING, -- Hash of Issuers public key
    serialNumber          CertificateSerialNumber }
```

2. OCSP 响应包

OCSP 响应包格式用 ASN.1 描述如下:

```

OCSPResponse ::= SEQUENCE {
    responseStatus      OCSPResponseStatus,
    responseBytes       [0] EXPLICIT ResponseBytes OPTIONAL }
OCSPResponseStatus ::= ENUMERATED {
    successful          (0), --Response has valid confirmations
    malformedRequest    (1), --Illegal confirmation request
    internalError       (2), --Internal error in issuer
    tryLater            (3), --Try again later
                        --(4) is not used
    sigRequired        (5), --Must sign the request
    unauthorized        (6)  --Request unauthorized    }
ResponseBytes ::= SEQUENCE {
    responseType      OBJECT IDENTIFIER,
    response          OCTET STRING }
id-pkix-ocsp        OBJECT IDENTIFIER ::= { id-ad-ocsp }
id-pkix-ocsp-basic  OBJECT IDENTIFIER ::= { id-pkix-ocsp 1 }
BasicOCSPResponse  ::= SEQUENCE {
    tbsResponseData    ResponseData,
    signatureAlgorithm  AlgorithmIdentifier,
    signature          BIT STRING,
    certs              [0] EXPLICIT SEQUENCE OF Certificate OPTIONAL }
ResponseData ::= SEQUENCE {
    version            [0] EXPLICIT Version DEFAULT v1,
    responderID        ResponderID,
    producedAt         GeneralizedTime,
    responses          SEQUENCE OF SingleResponse,
    responseExtensions [1] EXPLICIT Extensions OPTIONAL }
ResponderID ::= CHOICE {
    byName            [1] Name,
    byKey             [2] KeyHash }
KeyHash ::= OCTET STRING -- SHA-1 hash of responder's public key (excluding the tag and length
fields)
SingleResponse ::= SEQUENCE {
    certID            CertID,
    certStatus        CertStatus,
    thisUpdate        GeneralizedTime,
    nextUpdate        [0] EXPLICIT GeneralizedTime OPTIONAL,
    singleExtensions  [1] EXPLICIT Extensions OPTIONAL }
CertStatus ::= CHOICE {
    good              [0] IMPLICIT NULL,
    revoked           [1] IMPLICIT RevokedInfo,
    unknown           [2] IMPLICIT UnknownInfo }
RevokedInfo ::= SEQUENCE {

```

```

    revocationTime          GeneralizedTime,
    revocationReason [0]    EXPLICIT CRLReason OPTIONAL }
UnknownInfo ::= NULL -- this can be replaced with an enumeration

```

16.1.2 SOCSP

由于 OCSPP 请求包和响应包需要数字签名，从而导致其执行效率不高。为提高 OCSPP 的响应速度，采用 MAC 算法代替数字签名，形成简化版的 OCSPP，称作 SOCSP (Simple Online Certificate Status Protocol)。

1. SOCSP 请求包

SOCSP 请求包格式用 ASN.1 描述如下：

```

OCSPThinRequest ::= SEQUENCE {
    tbsThinRequest TBSThinRequest,      --请求信息
    mac            MacData              --请求信息的 MAC 值
}
TBSThinRequest ::= SEQUENCE {
    version [0] EXPLICIT Version DEFAULT v1,  --版本号
    random  BIT STRING,                      --随机数
    serialList SEQUENCE OF CertificateSerialNumber --请求查询证书的序列号
}
MacData ::= SEQUENCE {
    mac            DigestInfo,
    macSalt        OCTET STRING,
    iterations     INTEGER DEFAULT 1        --默认为 1, HASH 的反复次数
}
DigestInfo ::= SEQUENCE {
    digestAlgorithm DigestAlgorithmIdentifier,
    digest           Digest
}
Digest ::= OCTET STRING

```

与 OCSPP 不同，SOCSP 的请求数据需要进行基于对称密钥的 MAC 计算而不是基于公钥对的数字签名。为了提高系统的效率，SOCSP 没有设置扩展，所以用随机数来防止重放攻击，随机数以 random 字段出现在请求数据中，random 的长度 ≥ 128 位。

MAC 计算是基于 TBSThinRequest 结构的 DER 编码，加上共享密钥与 macSalt 的模 2 加进行的，macSalt 保证相同数据 MAC 的结果不相同。MAC 算法由 digestAlgorithm 标识，支持符合国家规定的 HMAC 算法。

2. SOCSP 响应包

SOCSP 响应包格式用 ASN.1 描述如下：

```

OCSPThinResponse ::=SEQUENCE {
    tbsThinResponse TBSThinResponse,      --响应数据
    Mac              MacData              --响应信息的 MAC 值
}

```



```

}
TBSThinResponse ::= SEQUENCE {
    version          [0] EXPLICIT Version DEFAULT v1,          --版本号
    random           BIT STRING,                                --随机数
    thinResponseStatus SEQUENCE OF OCSPThinResponseStatus --响应状态
}
OCSPThinResponseStatus ::= ENUMERATED {
    successful          (0), --Response has valid confirmations --响应被有效确认
    malformedRequest    (1), --Illegal confirmation request      --不完整的请求
    internalError       (2), --Internal error in issuer          --内部错误
    unauthorized        (3), --Request unauthorized              --未授权
}

```

从响应数据定义来看，响应包含了两个信息：响应信息和对响应信息进行 MAC 计算的值。响应数据包含：版本号、随机数、响应状态。随机数 `random` 与请求中的 `random` 相同；响应状态有 4 种：响应被有效确认、不完整的请求、内部错误、未授权。`MacData` 数定义与请求中一致。

16.2 CRL 服务

CRL 是证书作废列表，也称作黑名单。CA 系统通过 CRL 机制定期对外发布已作废（或已冻结）的证书序列号列表。用户端将最新的 CRL 下载到本地，通过解析 CRL 就可获得已作废证书的序列号、作废原因及作废时间等。

CRL 包含每个已作废证书的序列号、作废原因、作废时间等信息，但不含证书的具体内容。CA 系统定期生成新的 CRL，并将已过期证书从 CRL 中删除。

CRL 可以发布到 LDAP 中，也可以文件形式发布到网站上。

IETF RFC 3280 规定了 X.509 CRL 的格式，包括基本域组成、CRL 内容和扩展项。

16.2.1 基本域组成（CertificateList）

CRL 由 3 个域组成，具体见表 16-1。

表 16-1 CRL 基本域组成

分类	标识	说明
CRL 内容 (待签名)	tbsCertList	包含签发者信息、签发时间、已作废证书等
签名算法	signatureAlgorithm	包括摘要算法和公钥算法，如 sha1WithRSAEncryption，由算法标识和算法参数组成
签名值	signatureValue	使用签名算法，对 CRL 内容 tbsCertList 进行签名后的结果

CRL 基本域格式用 ASN.1 描述如下：

```

CertificateList ::= SEQUENCE {
    tbsCertList      TBSCertList,
    signatureAlgorithm AlgorithmIdentifier,

```

```

signatureValue      BIT STRING  }
AlgorithmIdentifier ::= SEQUENCE {
    Algorithm          OBJECT IDENTIFIER,
    Parameters         ANY DEFINED BY algorithm OPTIONAL  }

```

16.2.2 CRL 内容 (tbsCertList)

CRL 内容见表 16-2。

表 16-2 CRL 内容

分类	标识	说明
版本号	version	用于区分证书格式版本，最新版本为 v2
签名算法	signature	必须与基本域中的签名算法相同
CRL 签发者	issuer	用于区分 CRL 签发者，包含 CRL 签发者身份信息
本次签发时间	thisUpdate	本次 CRL 签发时间
下次签发时间	nextUpdate	下次 CRL 签发时间
作废证书集	revokedCertificates	包含已作废证书相关信息
扩展项	crlExtensions	包含其他可扩展信息

CRL 内容格式用 ASN.1 描述如下：

```

TBSCertList ::= SEQUENCE {
    version      Version OPTIONAL,      -- if present, MUST be v2
    signature     AlgorithmIdentifier,
    issuer        Name,
    thisUpdate     Time,
    nextUpdate     Time OPTIONAL,
    revokedCertificates SEQUENCE OF RevokedCertificate OPTIONAL,
    crlExtensions  [0] EXPLICIT Extensions OPTIONAL
    -- if present, MUST be v2
}

```

1. 版本号 version

version 用于区分 CRL 格式版本，最新版本为 v2。当使用扩展项 crlExtensions 时，version=v2。当 revokedCertificates 使用 crlEntryExtensions 属性时，version=v2。

version 格式用 ASN.1 描述如下：

```
Version ::= INTEGER { v1(0), v2(1) }
```

2. 签名算法 signature

signature 必须与基本域中签名算法相同，即：signature= CertificateList→signatureAlgorithm。

signature 格式用 ASN.1 描述如下：

```

AlgorithmIdentifier ::= SEQUENCE {
    Algorithm          OBJECT IDENTIFIER,
    Parameters         ANY DEFINED BY algorithm OPTIONAL  }

```

3. CRL 签发者 issuer

issuer 用于区分 CRL 签发者，必须包含一个 X.500 DN 项。DN 是 Distinguished Name 的缩写，表示可识别的名称，且 DN 项被定义为 X.501 规范中的 Name 类型。

CRL 签发者编码规则与证书签发者相同（具体参见“第 9 章”）。

issuer 格式用 ASN.1 描述如下：

```

Name ::= CHOICE {
    RDNSequence
}
RDNSequence ::= SEQUENCE OF RelativeDistinguishedName
RelativeDistinguishedName ::= SET OF AttributeTypeAndValue
AttributeTypeAndValue ::= SEQUENCE {
    type      AttributeType,
    value     AttributeValue
}
AttributeType ::= OBJECT IDENTIFIER
AttributeValue ::= ANY DEFINED BY AttributeType

```

4. 本次签发时间 thisUpdate 和下次签发时间 nextUpdate

thisUpdate 表示当前 CRL 的签发时间。

nextUpdate 表示下次 CRL 的最晚签发时间，即：下次 CRL 的实际签发时间可以早于 nextUpdate，但不能晚于 nextUpdate。同时，nextUpdate 不能早于已有 CRL 的签发时间。

thisUpdate 和 nextUpdate 均可以采用两种格式：UTCTime 和 GeneralizedTime。UTCTime 用 2 位数字表示年份，GeneralizedTime 用 4 位数字表示年份。2049 年以前的时间可采用 UTCTime 格式，但 2050 年及以后的时间必须采用 GeneralizedTime 格式。当采用 UTCTime 格式时，如 2 位年份数字 YY 小于 50 时，则年份应该解释为 20YY 年；当 YY 大于或等于 50 时，年份应该解释为 19YY 年。

5. 作废证书集 revokedCertificates

revokedCertificates 包含所有已作废证书的相关信息。当没有作废证书时，该字段不应出现。

revokedCertificates 格式用 ASN.1 描述如下：

```

revokedCertificates SEQUENCE OF RevokedCertificate OPTIONAL,
RevokedCertificate ::= SEQUENCE {
    userCertificate      CertificateSerialNumber,
    revocationDate       Time,
    crlEntryExtensions   Extensions OPTIONAL
                        -- if present, MUST be v2
}
CertificateSerialNumber ::= INTEGER

```

其中，revokedCertificates 由多个 CRL 条目组成。每个 CRL 条目包含单个作废证书的信息，其格式定义为 RevokedCertificate 类型，包括证书序列号 userCertificate、作废时间 revocationDate 和 CRL 条目扩展项 crlEntryExtensions。

6. 扩展项 crlExtensions 和 crlEntryExtensions

crlExtensions 用于 CRL 信息扩展，可包含多项扩展信息。

crlEntryExtensions 用于 CRL 条目信息扩展，可包含多项扩展信息。

crlExtensions 和 crlEntryExtensions 格式用 ASN.1 描述如下：

```
Extensions ::= SEQUENCE SIZE (1..MAX) OF Extension
Extension ::= SEQUENCE {
    extnID      OBJECT IDENTIFIER,
    critical    BOOLEAN DEFAULT FALSE,
    extnValue   OCTET STRING }
```

crlExtensions 和 crlEntryExtensions 只能出现在版本 v2 格式中。

每个扩展项都可设置为关键项（critical=TRUE）或非关键项（critical=FALSE）。如果遇到未知的关键扩展项，则必须拒绝该 CRL；如果遇到未知的非关键扩展项，可以忽略该扩展项。

每个扩展项由一个 OID 和一个 ASN.1 结构组成。OID 赋值给 extnID，ASN.1 编码后的结构赋值给 extnValue。

16.2.3 CRL 扩展项 crlExtensions

CRL 扩展项见表 16-3。

表 16-3 CRL 扩展项

/	扩展项	OID	是否关键项	说明
1	AuthorityKeyIdentifier	id-ce 35	FALSE	CRL 签发者密钥标识
2	IssuerAltName	id-ce 18	FALSE	CRL 签发者别名
3	CRLNumber	id-ce 20	FALSE	CRL 编号
4	DeltaCRLIndicator	id-ce 27	TRUE	增量 CRL 指示器
5	IssuingDistributionPoint	id-ce 28	TRUE	CRL 发布点
6	FreshestCRL	id-ce 46	FALSE	最新 CRL 或增量 CRL

注：id-ce OBJECT IDENTIFIER ::= { joint-iso-ccitt(2) ds(5) 29 }

1. authorityKeyIdentifier

authorityKeyIdentifier 扩展项用于区分 CRL 签发者的公钥。当 CRL 签发者拥有多个公私钥对用于签发 CRL 时，必须使用该扩展项。

该扩展项必须设置为非关键项（critical=FALSE）。

authorityKeyIdentifier 格式用 ASN.1 描述如下：

```
id-ce-authorityKeyIdentifier OBJECT IDENTIFIER ::= { id-ce 35 }
AuthorityKeyIdentifier ::= SEQUENCE {
    keyIdentifier      [0] KeyIdentifier      OPTIONAL,
    authorityCertIssuer [1] GeneralNames      OPTIONAL,
    authorityCertSerialNumber [2] CertificateSerialNumber OPTIONAL }
KeyIdentifier ::= OCTET STRING
```

authorityKeyIdentifier 基于 CRL 签发者证书中的内容生成，主要有两种生成方式：基于 subjectKeyIdentifier、基于 issuer 和 serialNumber。当基于 subjectKeyIdentifier 生成时，

keyIdentifier 通常等于 CRL 签发者证书中的 subjectKeyIdentifier。

2. issuerAltName

issuerAltName 扩展项表示 CRL 签发者的别名, 可包含多个。别名形式包括电子邮箱、DNS 名称、IP 地址、URI 等, 其中 DNS 名称也可以使用 issuer 中的 DN 项 domainComponent 表示。该扩展项必须设置为非关键项 (critical=FALSE)。

issuerAltName 格式用 ASN.1 描述如下:

```
id-ce-issuerAltName OBJECT IDENTIFIER ::= { id-ce 18 }
IssuerAltName ::= GeneralNames
GeneralNames ::= SEQUENCE SIZE (1..MAX) OF GeneralName
GeneralName ::= CHOICE {
    otherName                [0]    OtherName,
    rfc822Name                [1]    IA5String,
    dNSName                   [2]    IA5String,
    x400Address                [3]    ORAddress,
    directoryName              [4]    Name,
    ediPartyName               [5]    EDIPartyName,
    uniformResourceIdentifier  [6]    IA5String,
    iPAddress                  [7]    OCTET STRING,
    registeredID               [8]    OBJECT IDENTIFIER }
```

3. cRLNumber

cRLNumber 扩展项表示当前 CRL 的编号, 采用顺序递增的整数表示 (monotonically increasing sequence number), 用于快速区分不同的 CRL。cRLNumber 同时支持全量 CRL (包含所有作废证书) 和增量 CRL (只包含新作废证书)。

该扩展项必须设置为非关键项 (critical=FALSE)。

cRLNumber 格式用 ASN.1 描述如下:

```
id-ce-cRLNumber OBJECT IDENTIFIER ::= { id-ce 20 }
CRLNumber ::= INTEGER (0..MAX)
```

如果 CRL 签发者既生成全量 CRL, 也生成增量 CRL, 则这两类 CRL 必须统一编号 (share one numbering sequence), 即不允许独立编号。如果一个全量 CRL 和一个增量 CRL 同时生成, 则它们的 cRLNumber 必须相同, 且必须包含相同的作废证书集, 即该增量 CRL 与上一个全量 CRL 合并后, 与该全量 CRL 必须包含相同的作废证书集。

如果 CRL 签发者在不同的时间生成 2 个 CRL (2 个全量 CRL, 2 个增量 CRL, 或者 1 个全量 CRL 和 1 个增量 CRL), 则这 2 个 CRL 必须使用不同的 cRLNumber。也就是说, 如果两个 CRL 中的 thisUpdate 不相同, 则其 cRLNumber 也不能相同。

4. deltaCRLIndicator

deltaCRLIndicator 扩展项表示增量 CRL 指示器, 若存在, 则说明当前 CRL 是增量 CRL。增量 CRL 只包含最新的作废证书 (上次发布 CRL 后), 并不包含所有的作废证书。在有些环境下使用增量 CRL, 可以显著降低网络负载和处理时间。增量 CRL 通常比全量 CRL 要小, 因此应用系统获取增量 CRL 将比获取全量 CRL 消耗更少的网络带宽。本地保存作废

证书集时，建议不使用 CRL 结构而采用其他格式，因为获取增量 CRL 后，可以将其包含的新作废证书很方便地增加到本地保存的已有作废证书集中。

对于增量 CRL，必须包含该扩展项，且该扩展项必须设置为关键项（critical=TRUE）。

cRLNumber 格式用 ASN.1 描述如下：

```
id-ce-deltaCRLIndicator OBJECT IDENTIFIER ::= { id-ce 27 }
BaseCRLNumber ::= CRLNumber
```

其中，BaseCRLNumber 表示全量 CRL 的编号，该全量 CRL 是当前增量 CRL 的起点。也就是说，当前增量 CRL 与该全量 CRL 合并后，则包含所有的作废证书。

一个全量 CRL 和一个增量 CRL 允许合并，必须满足以下 4 个条件：

- ① 全量 CRL 和增量 CRL 具有相同的 CRL 签发者。
- ② 全量 CRL 和增量 CRL 具有相同的证书范围（scope）。只要满足其中一个条件，就可以认为两个 CRL 具有相同的证书范围。条件一：两个 CRL 均不包含 issuingDistributionPoint 扩展项；条件二：两个 CRL 的 issuingDistributionPoint 扩展项内容完全相同。

③ 全量 CRL 的 CRLNumber 等于或大于增量 CRL 的 BaseCRLNumber。也就是说，该全量 CRL 包含编号为 BaseCRLNumber 的全量 CRL 中所有作废证书。

④ 全量 CRL 的 CRLNumber 小于增量 CRL 的 CRLNumber。

5. issuingDistributionPoint

issuingDistributionPoint 扩展项表示 CRL 发布点和证书范围。通过该扩展项可显示出当前 CRL 的证书覆盖范围，如只覆盖终端证书、CA 证书、属性证书，或只覆盖特定作废原因的证书。CRL 由 CRL 签发者的私钥进行签名，但 CRL 发布点不需要自己的公私钥对。如果 CRL 存储在 X.500 目录服务器中，则应该保存到“CRL 发布点”对应的目录条目中，不应保存到“CRL 签发者”对应的目录条目中。

该扩展项必须设置为关键项（critical=TRUE）。

issuingDistributionPoint 格式用 ASN.1 描述如下：

```
id-ce-issuingDistributionPoint OBJECT IDENTIFIER ::= { id-ce 28 }
issuingDistributionPoint ::= SEQUENCE {
    distributionPoint          [0] DistributionPointName OPTIONAL,
    onlyContainsUserCerts     [1] BOOLEAN DEFAULT FALSE,
    onlyContainsCACerts       [2] BOOLEAN DEFAULT FALSE,
    onlySomeReasons           [3] ReasonFlags OPTIONAL,
    indirectCRL               [4] BOOLEAN DEFAULT FALSE,
    onlyContainsAttributeCerts [5] BOOLEAN DEFAULT FALSE }
```

其中，作废原因必须包含在 onlySomeReasons 字段中。如果 onlySomeReasons 没有出现，CRL 发布点必须包含所有作废原因的作废信息。例如，作废原因为 keyCompromise(1)、cACompromise(2)、aACompromise(8) 的作废证书可以在 A 发布点，其他作废证书可以在 B 发布点。

如果 distributionPoint 存在且包含一个 URI，则该 URI 必须链接到最新的 CRL，且只能使用绝对地址，必须指定主机名称，访问方式可以是 ftp、http、mail 和 ldap。

如果 CRL 签发者与证书签发者不相同，则 indirectCRL 必须设置为 TRUE；应在 CRL 条目扩展项 CertificateIssuer 中表明证书签发者。

6. freshestCRL (Delta CRL Distribution Point)

freshestCRL 扩展项用于确定如何获取当前全量 CRL 对应的增量 CRL 信息。该扩展项不能出现在增量 CRL 中。

该扩展项应该设置为非关键项 (critical=FALSE)。

freshestCRL 格式用 ASN.1 描述如下：

```
id-ce-freshestCRL OBJECT IDENTIFIER ::= { id-ce 46 }
FreshestCRL ::= CRLDistributionPoints
CRLDistributionPoints ::= SEQUENCE SIZE (1..MAX) OF DistributionPoint
DistributionPoint ::= SEQUENCE {
    distributionPoint      [0]      DistributionPointName OPTIONAL,
    reasons                [1]      ReasonFlags OPTIONAL,
    cRLIssuer              [2]      GeneralNames OPTIONAL }
DistributionPointName ::= CHOICE {
    fullName               [0]      GeneralNames,
    nameRelativeToCRLIssuer [1]      RelativeDistinguishedName }
```

其中，只有 distributionPoint 有效；reasons 和 cRLIssuer 无效，应忽略。

16.2.4 CRL 条目扩展项 crlEntryExtensions

CRL 条目扩展项见表 16-4。

表 16-4 CRL 条目扩展项

/	扩展项	OID	是否关键项	说明
1	ReasonCode	id-ce 21	FALSE	作废原因代码
2	HoldInstructionCode	id-ce 23	FALSE	冻结指令代码
3	InvalidityDate	id-ce 24	FALSE	证书无效日期
4	CertificateIssuer	id-ce 29	TRUE	证书签发者

注：id-ce OBJECT IDENTIFIER ::= { joint-iso-ccitt(2) ds(5) 29 }

1. reasonCode

reasonCode 扩展项表示证书作废原因。

该扩展项应该设置为非关键项 (critical=FALSE)。

reasonCode 格式用 ASN.1 描述如下：

```
id-ce-cRLReason OBJECT IDENTIFIER ::= { id-ce 21 }
reasonCode ::= { CRLReason }
CRLReason ::= ENUMERATED {
    unspecified            (0),
    keyCompromise          (1),  --表示密钥泄露
    cACompromise           (2),  --表示 CA 泄露
```

affiliationChanged	(3),	--表示关系变更
superseded	(4),	--表示废弃
cessationOfOperation	(5),	--表示操作中止
certificateHold	(6),	--表示证书冻结
removeFromCRL	(8),	--表示从 CRL 删除
privilegeWithdrawn	(9),	--表示权限撤销
aACompromise	(10)	--表示 AA 泄露
}		

2. holdInstructionCode

holdInstructionCode 扩展项表示冻结指令代码。当碰到作废原因为 certificateHold 的证书时，应按照该代码采取相应的行动。

该扩展项应该设置为非关键项（critical=FALSE）。

holdInstructionCode 格式用 ASN.1 描述如下：

```
id-ce-holdInstructionCode OBJECT IDENTIFIER ::= { id-ce 23 }
holdInstructionCode ::= OBJECT IDENTIFIER
```

常用的指令代码用 ASN.1 描述如下：

```
holdInstruction    OBJECT IDENTIFIER ::=
    { iso(1) member-body(2) us(840) x9-57(10040) 2 }
id-holdinstruction-none    OBJECT IDENTIFIER ::= {holdInstruction 1}
id-holdinstruction-callissuer    OBJECT IDENTIFIER ::= {holdInstruction 2}
id-holdinstruction-reject OBJECT IDENTIFIER ::= {holdInstruction 3}
```

其中，id-holdinstruction-callissuer 表示联系证书签发者或拒绝该证书。id-holdinstruction-reject 表示拒绝该证书。id-holdinstruction-none 表示不采取任何行动，与不包含该扩展项表示相同含义。

3. invalidityDate

invalidityDate 扩展项表示知道或怀疑私钥泄露或证书无效的时间。该时间可能早于 CRL 条目的作废时间。

该扩展项应该设置为非关键项（critical=FALSE）。

invalidityDate 格式用 ASN.1 描述如下：

```
id-ce-invalidityDate OBJECT IDENTIFIER ::= { id-ce 24 }
invalidityDate ::= GeneralizedTime
```

4. certificateIssuer

certificateIssuer 扩展项表示证书签发者。如果该 CRL 条目扩展项存在，则 CRL 扩展项 issuingDistributionPoint 的 indirectCRL 必须设置为 TRUE。如果第一个 CRL 条目的 certificateIssuer 扩展项不存在，则表示证书签发者与 CRL 签发者相同。如果某个 CRL 条目的 certificateIssuer 扩展项不存在，则表示该 CRL 条目的证书签发者与上一个 CRL 条目相同。

该扩展项应该设置为关键项（critical=TRUE）。

certificateIssuer 格式用 ASN.1 描述如下：


```
id-ce-certificateIssuer OBJECT IDENTIFIER ::= { id-ce 29 }
certificateIssuer ::= GeneralNames
```

16.3 LDAP 服务

LDAP 是 Lightweight Directory Access Protocol (轻型目录访问协议) 的缩写。CA 系统通过 LDAP 机制对外发布所有证书及 CRL。用户端可以通过 LDAP 协议访问 LDAP 服务器, 按需下载满足条件的证书和 CRL。

16.3.1 发布数字证书到 LDAP

1. 定义 LDAP schema

LDAP schema 是一个规则集, 定义了 LDAP 目录所应遵循的结构和规则, 它决定哪些信息可以存放在目录服务中, 以及在客户端和目录服务器查询交互中如何处理信息。目录服务器在存储新数据或修改现有数据时, 会检查数据是否满足 schema 规则。当客户端或服务端比较两个属性值时, 它们会使用 schema 规定的比较算法。

LDAP schema 主要由四个元素组成: 对象类 (objectClass)、属性 (attribute)、语法 (Syntax)、匹配规则 (Matching Rules)。

LDAP schema 定义后, 将形成 LDAP 目录树结构。例如, 针对 DN 项为 “c=?,st=?,o=?,ou=?,cn=?”, 定义 LDAP schema 后形成的 LDAP 目录树如图 16-1 所示。

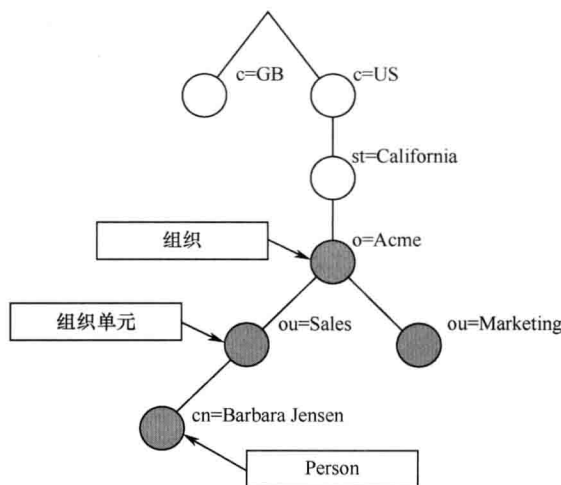


图 16-1 LDAP 目录树 (传统命名方式)

2. 添加数字证书条目到 LDAP

CA 系统签发每个数字证书后, 将实时或定期通过 API 方式将该数字证书发布到 LDAP 服务器。

CA 系统添加数字证书条目到 LDAP 的步骤主要包括:

- ① 打开 LDAP Server 连接。ldap_open() 返回连接句柄, 允许多个连接同时打开。
- ② 同 LDAP Server 进行身份认证。ldap_bind() 及相关函数支持多种不同的认证方法。

③ 执行 LDAP 条目添加操作并返回执行结果。ldap_add()和 ldap_add_s()用来添加 LDAP 目录条目。

④ 关闭 LDAP Server 连接。由 ldap_unbind()调用实现。

16.3.2 访问 LDAP 获取数字证书

1. 通过软件工具访问

向对方发送安全邮件时，使用对方的数字证书对邮件内容进行加密，因此邮件发送前需要获得对方的数字证书，很多电子邮件软件已经支持通过访问 LDAP 获得数字证书。

以 Foxmail 6.5 为例，对于每个联系人，可以通过其电子邮箱从 LDAP 查询并获得其数字证书。

进入“地址簿”→“每个卡片”→“属性”→“数字证书”界面后，单击“LDAP 搜索”按钮即可进行搜索，如图 16-2 所示。

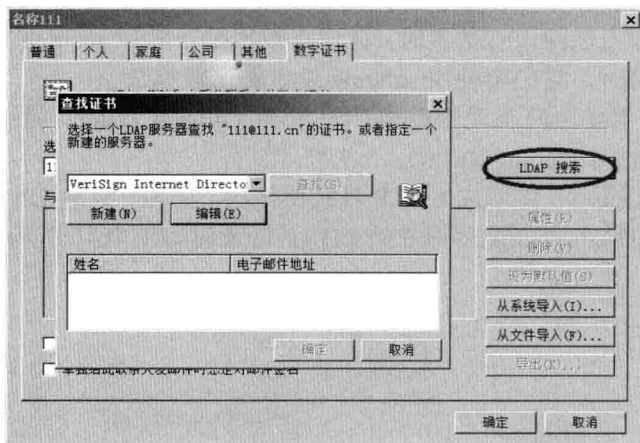


图 16-2 Foxmail 6.5 中 LDAP 搜索界面

通过“新增”或“编辑”按钮可对 LDAP 进行新增或修改，如图 16-3 所示。

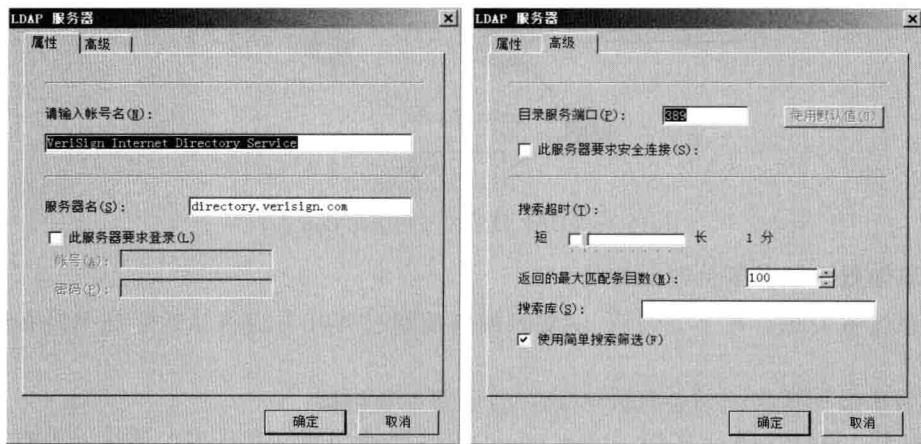


图 16-3 Foxmail 6.5 中 LDAP 设置界面

2. 通过 API 方式访问

应用程序通过 API 方式访问 LDAP 需要进行以下 4 个步骤：

- ① 打开 LDAP Server 连接。`ldap_open()`返回连接句柄，允许多个连接同时打开。
- ② 同 LDAP Server 进行身份认证。`ldap_bind()`及相关函数支持多种不同的认证方法。
- ③ 根据条件执行 LDAP 查询操作并获得查询结果。`ldap_search()`及相关函数可执行 ldap 查询操作并获取结果；返回结果可以由 `ldap_result2error()`、`ldap_first_entry()`、`ldap_next_entry()`解析。ldap 操作支持同步或异步执行。
- ④ 关闭 LDAP Server 连接。由 `ldap_unbind()`调用实现。

第 17 章 网络部署结构

17.1 运营型 CA

根据运营型 CA 的安全要求, CA 中心需要划分不同安全级别的安全域, 并将不同的模块部署在不同的安全域内。按照安全级别的高低, CA 中心的安全域主要包括:

- ① KM 区: 部署密钥管理系统。
- ② 核心区: 部署证书/CRL 签发系统、主 LDAP 系统、主 OCSP 系统等。
- ③ 管理区: 部署证书管理系统。
- ④ 服务区: 部署从 LDAP 系统、从 OCSP 系统、RA 系统等。

CA 中心网络部署结构如图 17-1 所示。

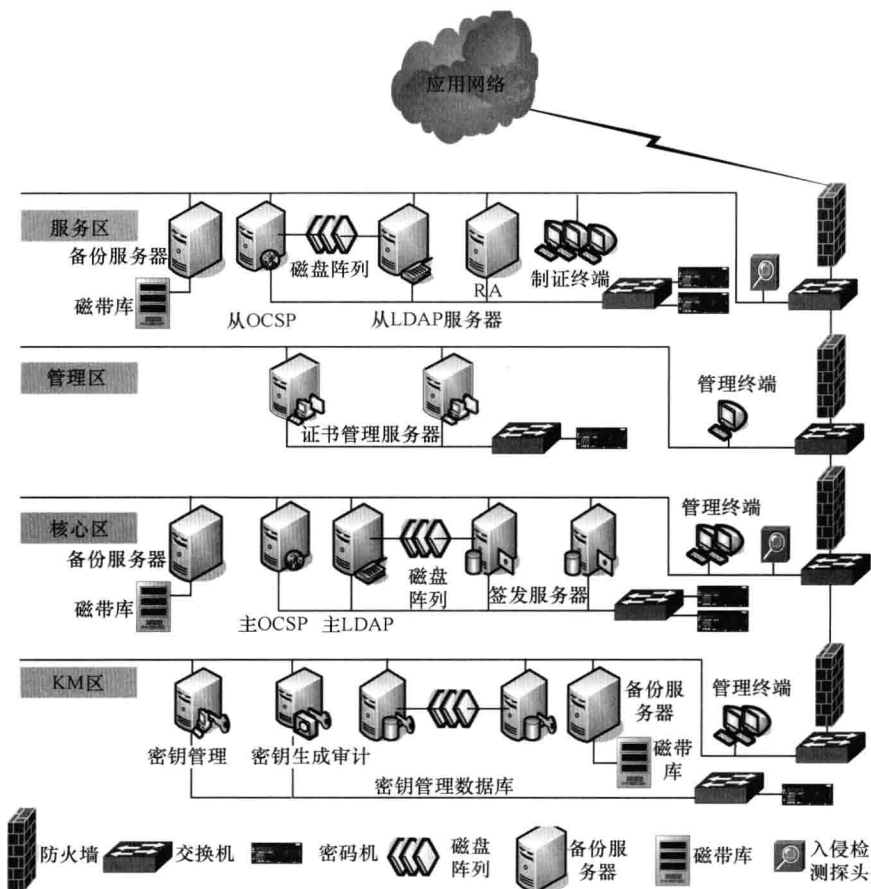


图 17-1 CA 中心网络部署结构

图中，KM 区内，密钥管理系统只接收来自核心区的证书/CRL 签发系统的服务请求。核心区内，证书/CRL 签发系统只接收来自管理区证书管理系统的请求；主 LDAP 系统和主 OCSP 系统都是单向通信，只定期将数据同步到服务区内的从 LDAP 系统和从 OCSP 系统。管理区内，证书管理系统只接收来自服务区 RA 系统的各种业务请求。服务区内，RA 系统负责面向直接用户，提供证书申请、身份审核和证书下载等业务服务。

各安全域之间通过防火墙进行边界保护，并部署 IDS、漏洞扫描、防病毒等安全系统进行网络安全和主机安全防护。

为确保业务的连续性，尽量降低或避免单点故障对运营服务的营销，可配置双网络链路、核心服务器双机热备、磁盘阵列等。为保证业务服务性能，从 LDAP 系统和从 OCSP 系统的硬件服务器性能要优于其他服务器。

当 RA 采取浏览器/服务器 (B/S) 模式时，可将服务区与管理区网络放在同一网段，网络部署结构可参考图 17-2。

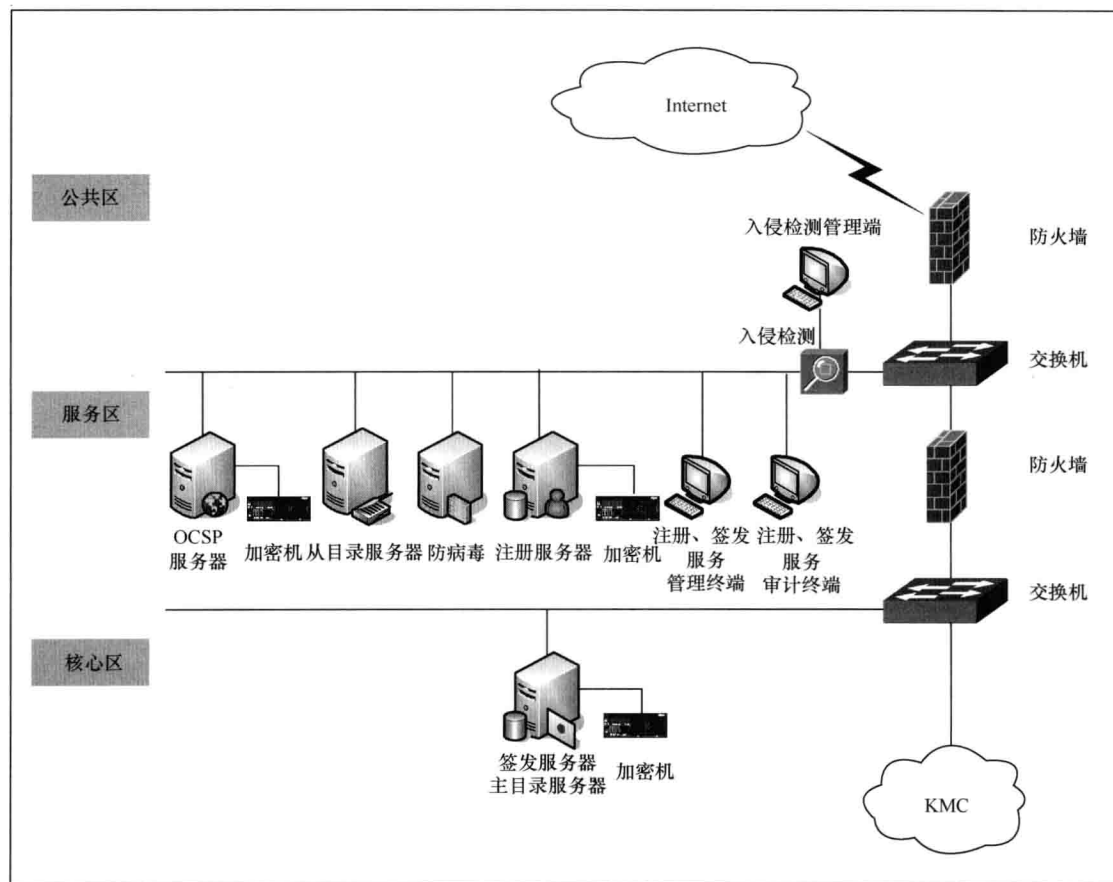


图 17-2 RA 采用 B/S 模式时 CA 的网络结构示意图

CA 与远程 RA 的连接时，网络部署结构可参考图 17-3。

KMC 与多个 CA 的网络连接时，网络部署结构可参考图 17-4。

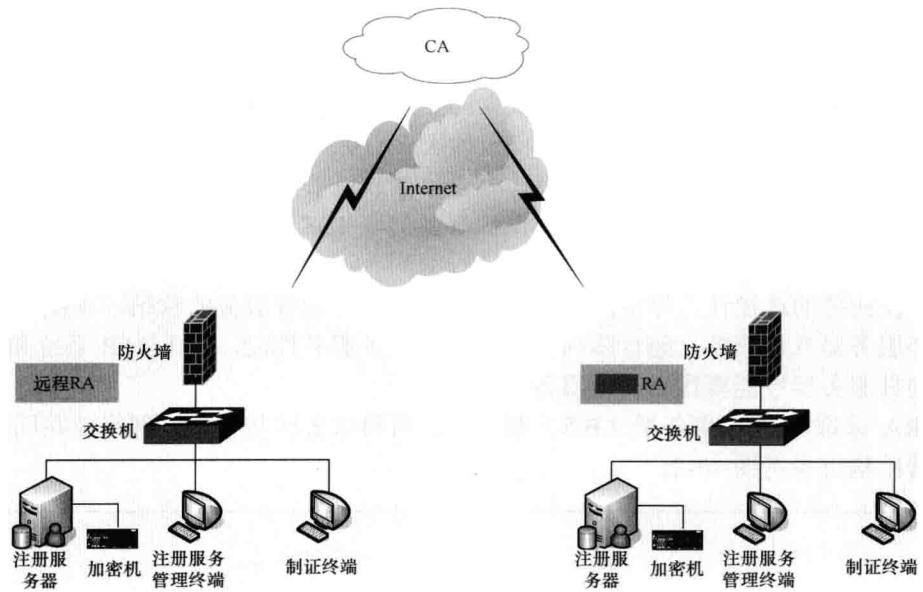


图 17-3 CA 与远程 RA 的连接示意图

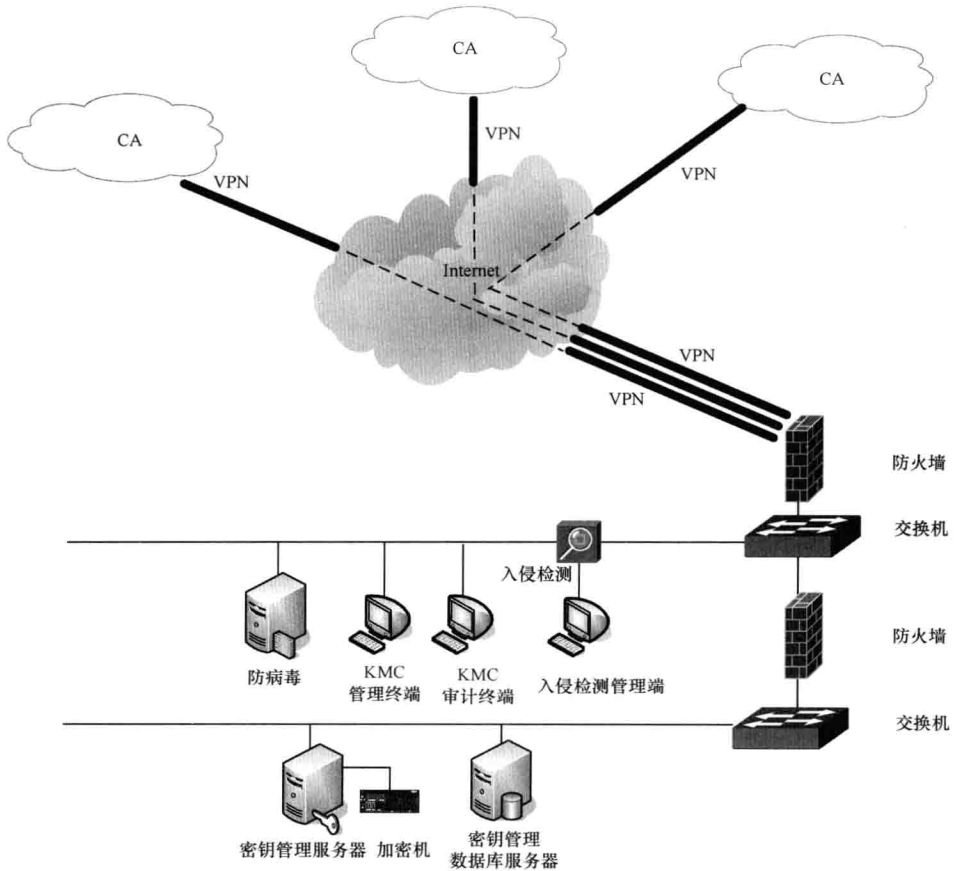


图 17-4 KMC 与多个 CA 的网络连接示意图

17.2 企业级 CA

企业级 CA 应该可以根据模块进行灵活部署, 通常分为几种模式: 双层标准模式、双层简化模式、单层单机模式和纯硬件模式。企业级 CA 模块组成可参考图 15-3。

17.2.1 双层标准模式

该模式是企业级 CA 的标准部署结构, 将核心层模块和服务层模块分别部署在 2 个不同的安全域中, 安全域之间采用防火墙和 VLAN 分隔保护。根 CA 签名模块只签发 CA 证书, 为保证安全性, 通过脱机方式实现。加密机通过硬件保护 CA 密钥对的安全性, 保证了证书签发的安全性。

RA 受理点的管理员和操作员通过互联网访问部署在中心机房的 RA 服务器; 证书用户通过互联网访问部署在中心机房的用户服务器。

CA 管理员、RA 管理员、RA 操作员、证书用户等, 无需配置特殊设备, 只采用普通 PC 就可以访问 CA 系统。

该模式下, 网络部署结构如图 17-5 所示。

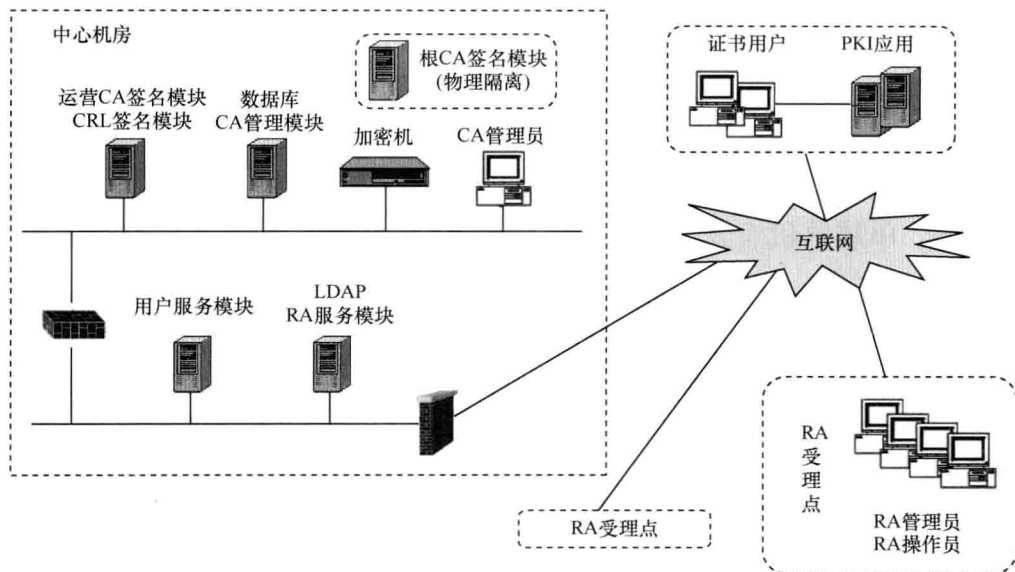


图 17-5 企业级 CA 双层标准模式网络部署结构

17.2.2 双层简化模式

该模式是企业级 CA 标准部署结构的简化模式, 将核心层模块和服务层模块分别部署在 2 个不同的安全域中, 安全域之间采用防火墙和 VLAN 分隔保护。根 CA 签名模块只签发 CA 证书, 为保证安全性, 通过脱机方式实现。加密机通过硬件保护 CA 密钥对的安全性, 保证了证书签发的安全性。为降低部署硬件成本, 核心层所有模块和服务层所有模块分别安装在单台硬件服务器上。

RA 受理点的管理员和操作员以及证书用户均通过互联网访问部署在中心机房的 RA/ 用户服务器。

CA 管理员、RA 管理员、RA 操作员、证书用户等，无需配置特殊设备，只采用普通 PC 就可以访问 CA 系统。

该模式下，网络部署结构如图 17-6 所示。

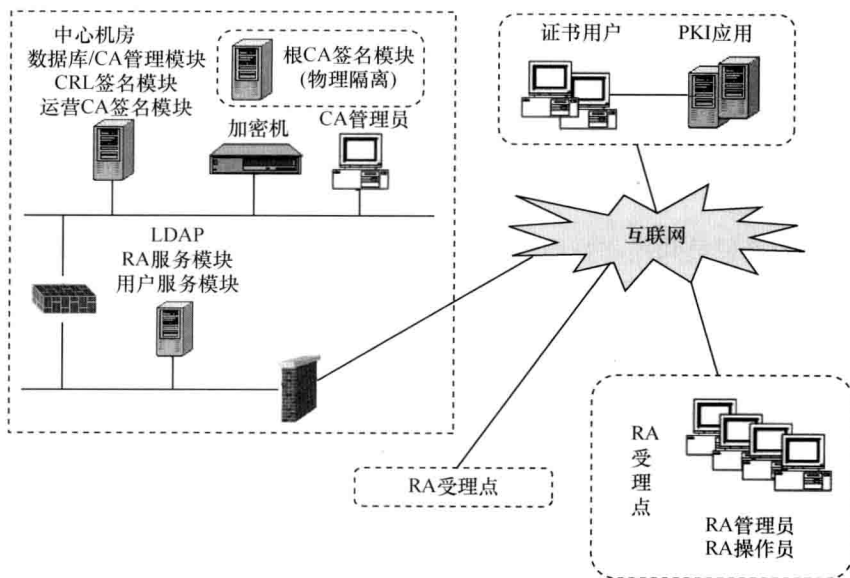


图 17-6 企业级 CA 双层简化模式网络部署结构

17.2.3 单层单机模式

该模式是企业级 CA 最简单的部署结构，为进一步降低硬件成本，不再划分安全域，将核心层和服务层所有模块均部署在同一台硬件服务器上。加密机通过硬件保护 CA 密钥对的安全性，保证了证书签发的安全性。

RA 受理点的管理员和操作员以及证书用户均通过互联网访问部署在中心机房的 RA/ 用户服务器。

CA 管理员、RA 管理员、RA 操作员、证书用户等，无需配置特殊设备，只采用普通 PC 就可以访问 CA 系统。

该模式下，网络部署结构如图 17-7 所示。

17.2.4 纯硬件模式

该模式属于“交钥匙”部署模式，无需用户提供任何设备，只需安装证书服务器设备即可，采用内置式加密卡保护关键密钥的安全性。核心层和服务层所有模块均部署在该证书服务器设备中。

RA 受理点的管理员和操作员以及证书用户均通过互联网访问部署在中心机房的证书服务器。

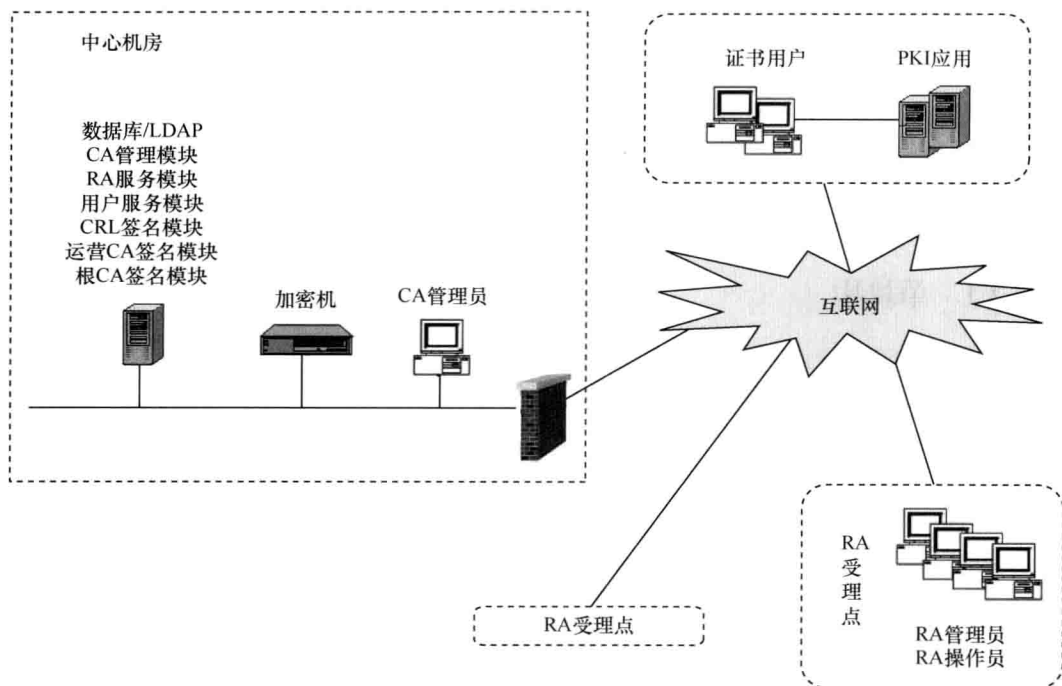


图 17-7 企业级 CA 单层单机模式网络部署结构

CA 管理员、RA 管理员、RA 操作员、证书用户等，无需配置特殊设备，只采用普通 PC 就可以访问 CA 系统。

该模式下，网络部署结构如图 17-8 所示。

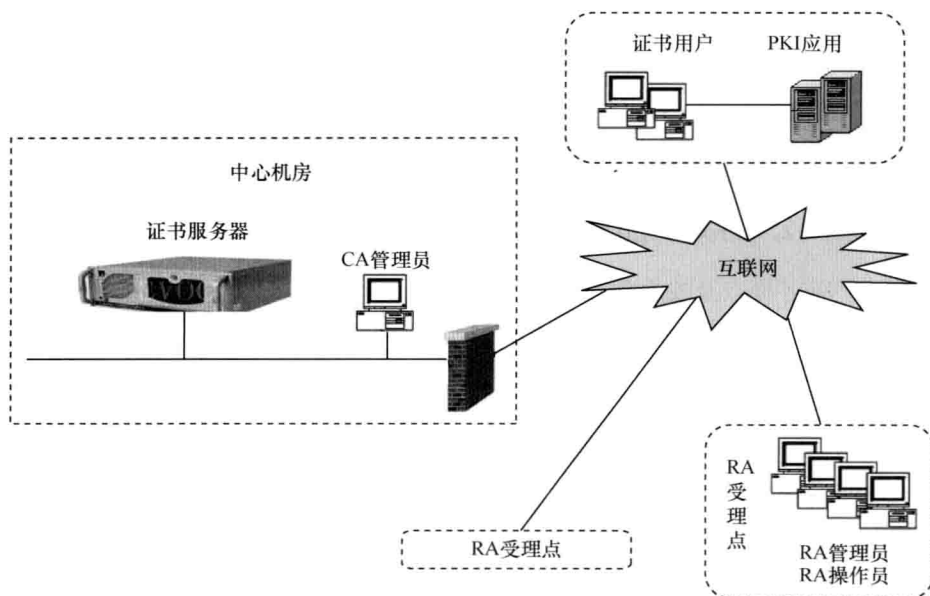


图 17-8 企业级 CA 纯硬件模式网络部署结构

17.3 按企业管理模式部署 CA

企业级 CA 应该可以根据企业管理模式进行灵活部署，通常分为几种模式：单机构、集团公司+集中部署+集中发证、集团公司+集中部署+分布发证、集团公司+两级部署+分布发证。

为了方便说明，本节将企业级 CA 所有功能模块及其网络部署结构抽象成证书服务器，具体部署结构需要根据实际需要选择 17.2 节中的合适模式。

17.3.1 单机构

该模式下，应用特点及安全需求主要包括：

① 没有独立的二级机构。

② 应用系统集中部署。

数字证书部署要点主要包括：

① 部署一套证书服务器。

② 由证书服务器为员工签发个人证书，数字证书及私钥保存在 USB Key 中。

③ 由证书服务器为应用系统签发设备/服务器证书。

④ 基于数字证书技术，保证了员工访问应用系统的安全性。

该模式下，网络部署结构如图 17-9 所示。

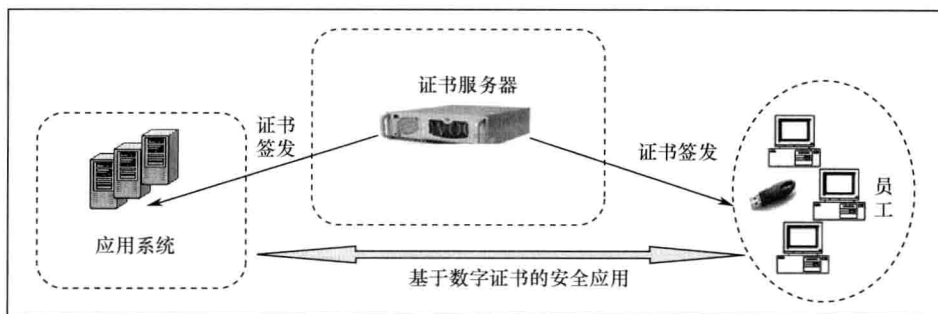


图 17-9 企业级 CA 单机构模式网络部署结构

17.3.2 集团公司+集中部署+集中发证

该模式下，应用特点及安全需求主要包括：

① 组织结构分为 2 级：集团总公司和分/子公司。

② 应用系统部分集中部署，部分分布部署。

③ 数字证书系统要求集中部署。

④ 数字证书发证业务要求集中发证。

数字证书部署要点主要包括：

① 在总公司部署一套证书服务器。

② 由证书服务器为总公司员工签发个人证书，数字证书及私钥保存在 USB Key 中。

③ 由证书服务器为总公司应用系统签发设备/服务器证书。

④ 基于数字证书技术，保证了总公司员工访问总公司应用系统的安全性。

- ⑤ 由证书服务器为分/子公司员工签发个人证书,数字证书及私钥保存在 USB Key 中。
- ⑥ 由证书服务器为分/子公司应用系统签发设备/服务器证书。
- ⑦ 基于数字证书技术,保证了分/子公司员工访问总公司应用系统和分/子公司应用系统的安全性。

该模式下,网络部署结构如图 17-10 所示。

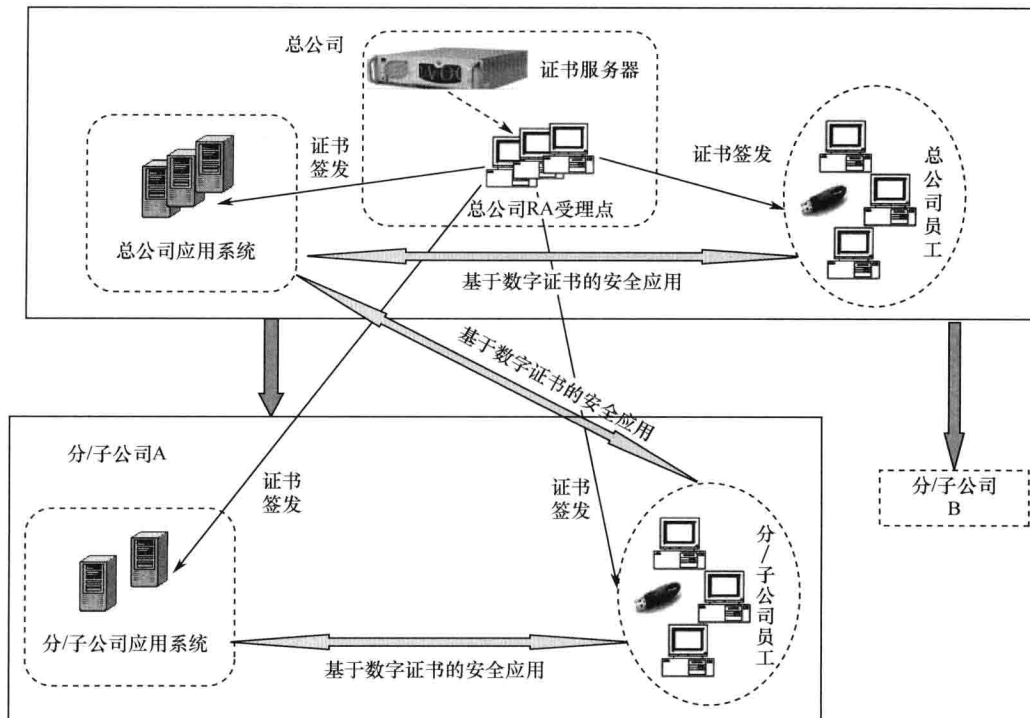


图 17-10 企业级 CA 集团模式 I 网络部署结构

17.3.3 集团公司+集中部署+分布发证

该模式下,应用特点及安全需求主要包括:

- ① 组织结构分为 2 级:集团总公司和分/子公司。
- ② 应用系统部分集中部署,部分分布部署。
- ③ 数字证书系统要求集中部署。
- ④ 数字证书发证业务要求分布发证。

数字证书部署要点主要包括:

- ① 在总公司部署一套证书服务器,并支持多个 RA 受理点。
- ② 由总公司 RA 受理点为总公司员工签发个人证书,数字证书及私钥保存在 USB Key 中。
- ③ 由总公司 RA 受理点为总公司应用系统签发设备/服务器证书。
- ④ 基于数字证书技术,保证了总公司员工访问总公司应用系统的安全性。
- ⑤ 由分/子公司 RA 受理点为分/子公司员工签发个人证书,数字证书及私钥保存在 USB Key 中。
- ⑥ 由分/子公司 RA 受理点为分/子公司应用系统签发设备/服务器证书。

⑦ 基于数字证书技术，保证了分/子公司员工访问总公司应用系统和分/子公司应用系统的安全性。

该模式下，网络部署结构如图 17-11 所示。

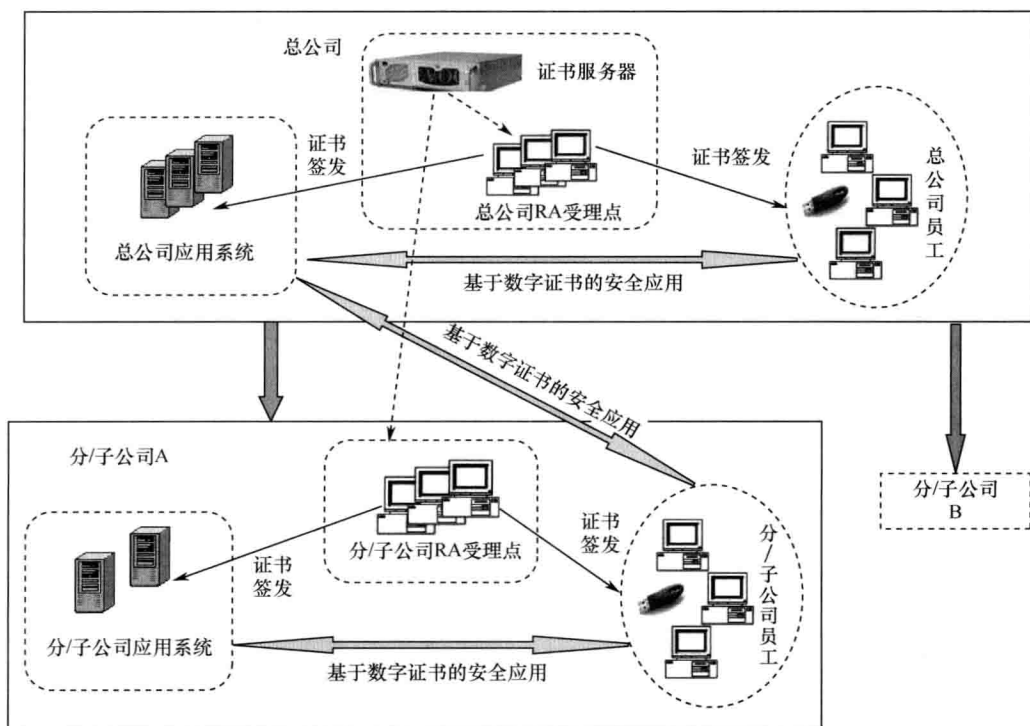


图 17-11 企业级 CA 集团模式 II 网络部署结构

17.3.4 集团公司+两级部署+分布发证

该模式下，应用特点及安全需求主要包括：

- ① 组织结构分为 2 级：集团总公司和分/子公司。
- ② 应用系统部分集中部署，部分分布部署。
- ③ 数字证书系统要求分布部署。
- ④ 数字证书发证业务要求分布发证。

数字证书部署要点主要包括：

- ① 在总公司部署一套根证书服务器。
- ② 在总公司部署一套证书服务器。
- ③ 由总公司证书服务器为总公司员工签发个人证书，数字证书及私钥保存在 USB Key 中。
- ④ 由总公司证书服务器为总公司应用系统签发设备/服务器证书。
- ⑤ 基于数字证书技术，保证了总公司员工访问总公司应用系统的安全性。
- ⑥ 在分/子公司部署一套证书服务器。
- ⑦ 由分/子公司证书服务器为分/子公司员工签发个人证书，数字证书及私钥保存在

USB Key 中。

⑧ 由分/子公司证书服务器为分/子公司应用系统签发设备/服务器证书。

⑨ 基于数字证书技术，保证了分/子公司员工访问总公司应用系统和分/子公司应用系统的安全性。

该模式下，网络部署结构如图 17-12 所示。

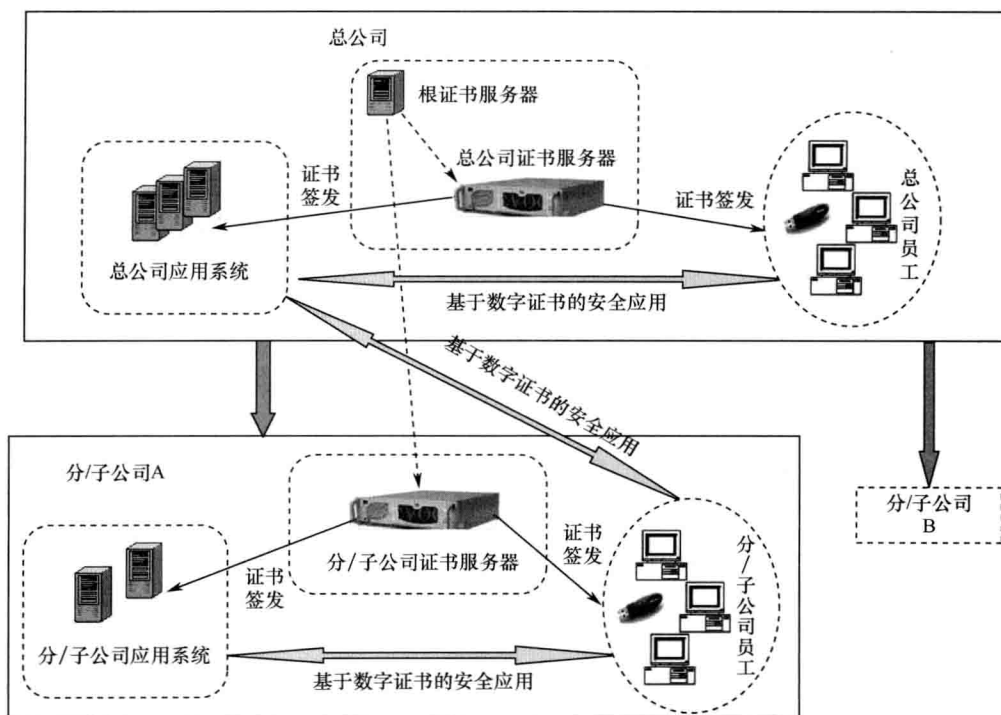


图 17-12 企业级 CA 集团模式 III 网络部署结构

第 18 章 实 验 三

18.1 OpenSSL CA 示例

18.1.1 简介

OpenSSL 是一个功能丰富的开源安全工具箱,它提供的主要功能有:多种软件算法(对称/非对称/摘要)、大数运算、非对称算法密钥生成、ASN.1 编解码库、时间戳、证书请求(PKCS10)编解码、数字证书/CRL 编解码、OCSP 协议、数字证书验证、PKCS7 标准实现、PKCS12 个人数字证书格式实现、SSL 协议实现(包括 SSLv2、SSLv3 和 TLSv1)等。OpenSSL 官方网址为 <http://www.openssl.org>。本章只涉及 OpenSSL 中的 CA 功能。

OpenSSL 采用 C 语言作为开发语言,这使得它具有优秀的跨平台性能,支持 Linux、UNIX、windows、Mac 等平台。

18.1.2 安装配置

本节以 32 位 Windows 7 操作系统环境为例进行安装配置说明,其他操作系统环境类似。从 Shining Light 网站 <http://slproweb.com/products/Win32OpenSSL.html> 下载编译好的安装软件包,如果读者感兴趣,可以下载源代码进行编译。在下载页面有多个版本的 OpenSSL,分别是 1.0.1g、1.0.0L、0.9.8y,每个版本号下对应 32 位和 64 位版本,根据自己的操作系统环境可以选择下载对应的版本。这里以 1.0.1g 的 32 位版本进行说明,下载链接说明如表 18-1 所示。

表 18-1 下载链接说明

文件	类型	说明
Win32 OpenSSL v1.0.1g Light	1MB 安装包	建议使用版本 OpenSSL v1.0.1g,按缺省条件编译
Visual C++ 2008 Redistributables	1.7MB 安装包	支持库,适合 Windows 2000 以后系统
Visual C++ 2008 Redistributables for Windows 9x/NT4	4.6MB 压缩包	支持 Windows 95/98/Me/NT4 操作系统

1. 安装 Visual C++ 2008 Redistributables

单击表 18-1 中的运行库“Visual C++ 2008 Redistributables”,浏览器会导航到微软网站,见图 18-1,单击“Download”进行下载。

下载完成后,双击该安装包。当出现安全警告对话框(因从网络下载,系统产生安全警告),提示“无法验证发布者,您确定要运行此软件吗?”时,单击“运行”按钮。当出现许可条款对话框时,勾选“我已阅读并接受许可条款”,单击“安装”按钮。其他安装界面按照提示选择缺省按钮即可完成安装。

安装完成后,会出现如图 18-2 所示画面,单击“完成”按钮,结束安装。

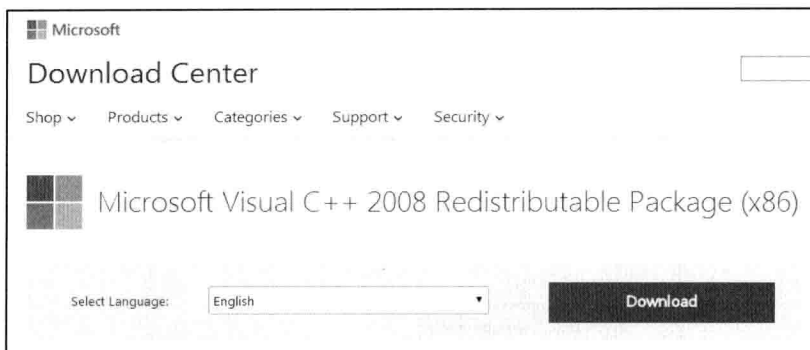


图 18-1 运行库支持包下载页

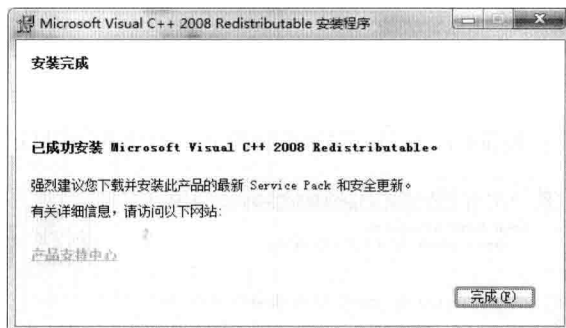


图 18-2 运行库安装完成界面

安装完成后，请重新启动操作系统。

2. 安装 Win32 OpenSSL v1.0.1g Light

单击表 18-1 中链接下载“Win32 OpenSSL v1.0.1g Light”软件包，下载完成后，双击安装包 Win32OpenSSL_Light-1_0_1g.exe。当出现安全警告对话框，提示“无法验证发布者，您确定要运行此软件吗？”时，单击“运行”按钮，进入安装过程。

当出现如图 18-3 所示画面时，表示系统缺少 Visual C++ 2008 Redistributables 运行库或该运行库安装不正确，请重新安装。

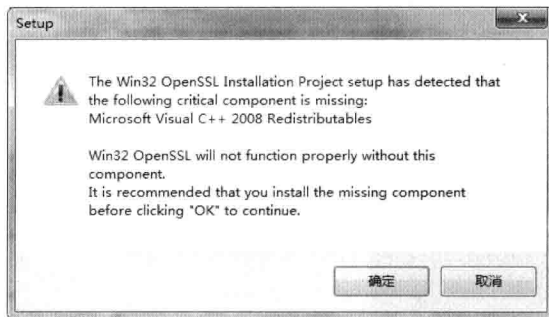


图 18-3 缺少运行库警告

当出现“License Agreement”对话框时，选择“I accept the agreement”，单击“Next”

按钮。

当出现如图 18-4 所示画面时，选择“The OpenSSL binaries (/bin) directory”，表示把 OpenSSL 动态库复制到 OpenSSL 的安装目录的 bin 目录下。

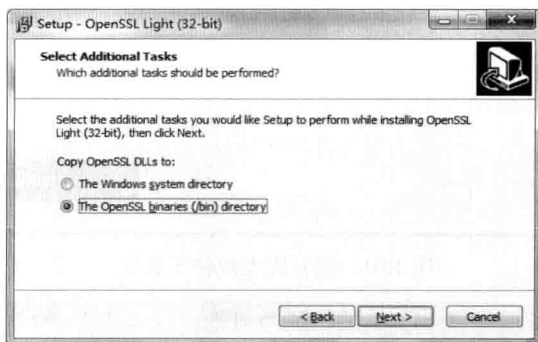


图 18-4 选择复制 OpenSSL 动态库位置

当出现如图 18-5 所示画面时，要求选择安装路径，本例使用“D:\var\OpenSSL-Win32”。

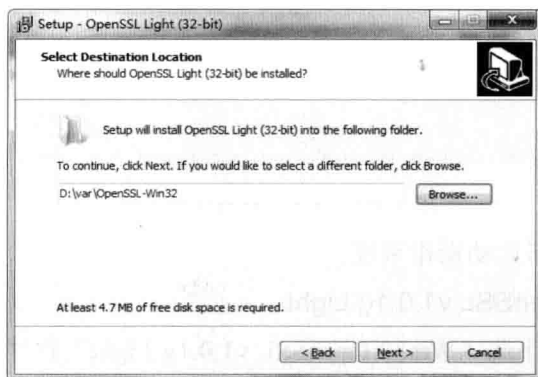


图 18-5 安装路径选择

安装完成后，会出现如图 18-6 所示画面，单击“Finish”按钮，结束安装。

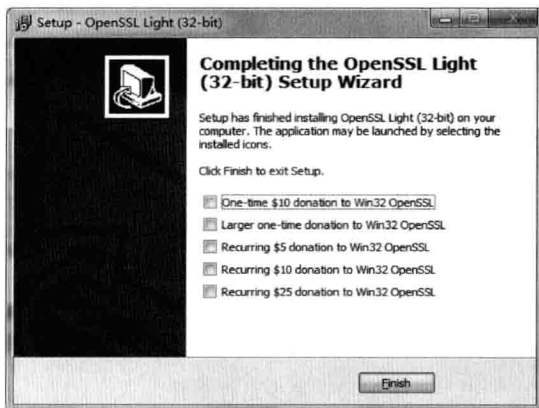


图 18-6 安装结束界面

3. 配置环境变量

打开安装后路径(本例为 D:\var\OpenSSL-Win32\bin), 安装后文件信息如图 18-7 所示。

名称	类型	大小
PEM	文件夹	
4758cca.dll	应用程序扩展	14 KB
aep.dll	应用程序扩展	13 KB
atalla.dll	应用程序扩展	12 KB
CA.pl	PL 文件	6 KB
capi.dll	应用程序扩展	23 KB
chil.dll	应用程序扩展	17 KB
cswift.dll	应用程序扩展	16 KB
FixSSL_9xNT4.bat	Windows 批处理...	2 KB
gmp.dll	应用程序扩展	7 KB
gost.dll	应用程序扩展	58 KB
libeay32.dll	应用程序扩展	1,150 KB
nuron.dll	应用程序扩展	11 KB
openssl.cfg	CFG 文件	11 KB
openssl.exe	应用程序	385 KB
padlock.dll	应用程序扩展	12 KB
ssleay32.dll	应用程序扩展	264 KB
sureware.dll	应用程序扩展	17 KB
ubsec.dll	应用程序扩展	15 KB

图 18-7 安装后文件信息

其中最常用的有 openssl.exe (命令行工具)、libeay32.dll (算法库实现)、ssleay32.dll (SSL/TLS 协议实现)、openssl.cfg (参数配置文件)。本例主要使用 openssl.exe 完成 CA 功能演示。

在安装目录下, 打开命令行窗口 (所有程序→附件→命令提示符), 执行命令“openssl.exe version”可查询版本信息, 显示如图 18-8 所示信息, 表示 OpenSSL 版本为 1.0.1g。(下文中“执行命令”均指在命令行窗口中输入该命令。)

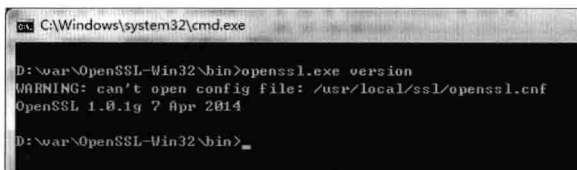


图 18-8 显示 OpenSSL 版本

在显示版本时, 出现“WARNING: can't open config file: /usr/local/ssl/openssl.cnf”消息, 这是因为缺省的配置文件/usr/local/ssl/openssl.cnf 不存在。可以通过环境变量进行更改。

设置环境变量 OPENSSL_CONF, 使其指向新的配置文件, 执行命令:

```
set OPENSSL_CONF=D:\var\OpenSSL-Win32\bin\openssl.cfg
```

然后再执行命令“openssl.exe version”, 显示如图 18-9 所示信息, 此时没有了警告信息。

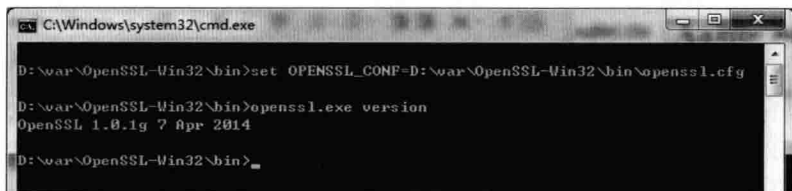


图 18-9 设置环境变量 OPENSSL_CONF 后查看版本

18.1.3 申请证书

1. 创建相关文件

在 D:\var\OpenSSL-Win32\bin 下创建文件夹 demoCA，并在 demoCA 下创建子文件夹：certs、crl、newcerts、private。

创建几个文本文件：

① index.txt。

② serial。在 serial 文件中写入初始证书序列号，可以设置为 AEF0（十六进制表示的初始证书序列号）。

③ crlnumber。在 crlnumber 中写入 CRL 序列号，可以设置为 01（十六进制表示）。

创建完成后结果如下所示：

```
-- demoCA/  
|-- certs/  
|-- crl/  
|-- newcerts/  
|-- private/  
|-- index.txt  
|-- serial  
|-- crlnumber
```

2. 创建 CA 证书

先用缺省配置文件产生 CA 证书和用户证书，此处 CA 证书为自签名证书。

创建的根 CA 证书的名字为 OpenSSL CA，执行命令：

```
openssl.exe req -x509 -newkey rsa:2048 -days 3660 -out .\demoCA\cacert.pem -outform PEM  
-keyout .\demoCA\private\cakey.pem -subj "/C=CN/CN=OpenSSL CA"
```

在执行过程中需要输入私钥保护口令两次，显示信息如图 18-10 所示。openssl.exe req 支持的参数可以通过执行命令“openssl.exe req help”查看。

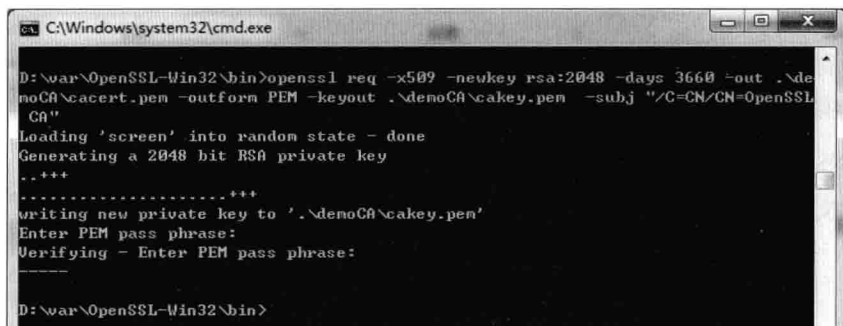


图 18-10 产生根证书的过程

上面命令的含义为：产生密钥长度为 2048 位的 RSA 密钥，证书的有效期为 3660 天，这是一个自签名证书，产生的证书输出为 demoCA 目录下的 cacert.pem，证书格式为 PEM，

产生的私钥为 demoCA 目录下的 cakey.pem，证书的通用名为 OpenSSL CA。

打开刚产生的证书，在 Windows 下显示如图 18-11 所示信息。

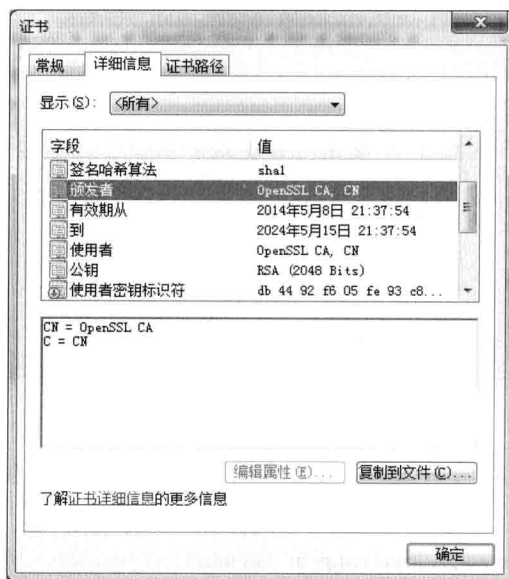


图 18-11 产生的 CA 证书信息

根证书产生后，把 cakey.pem 复制到.\demoCA\private\目录下，然后修改 openssl.cfg 文件，找到[CA_default]标志，具体信息如下：

```
[ CA_default ]
dir               = ./demoCA           # Where everything is kept
certs             = $dir/certs         # Where the issued certs are kept
crl_dir           = $dir/crl           # Where the issued crl are kept
database          = $dir/index.txt     # database index file.
#unique_subject   = no                # Set to 'no' to allow creation of
                                      # several ctificates with same subject.
new_certs_dir     = $dir/newcerts      # default place for new certs.
certificate       = $dir/cacert.pem    # The CA certificate
serial            = $dir/serial        # The current serial number
crlnumber         = $dir/crlnumber     # the current crl number
                                      # must be commented out to leave a V1 CRL
crl               = $dir/crl.pem       # The current CRL
private_key       = $dir/private/cakey.pem # The private key
RANDFILE          = $dir/private/.rand # private random number file
```

找到[CA_default]下的“policy = policy_match”并将其修改为“policy = policy_anything”。其他值采用缺省值。

3. 创建用户证书

在创建完 CA 证书并修改配置文件后，就可以创建用户证书了。先创建用户证书请求

文件，用户名字为 user-1，执行命令：

```
openssl.exe req -newkey rsa:1024 -days 3640 -keyout .\demoCA\private\user1key.pem -keyform PEM
-out .\demoCA\user1req.pem -outform PEM -nodes -subj "/C=CN/CN=user-1"
```

显示信息如图 18-12 所示。

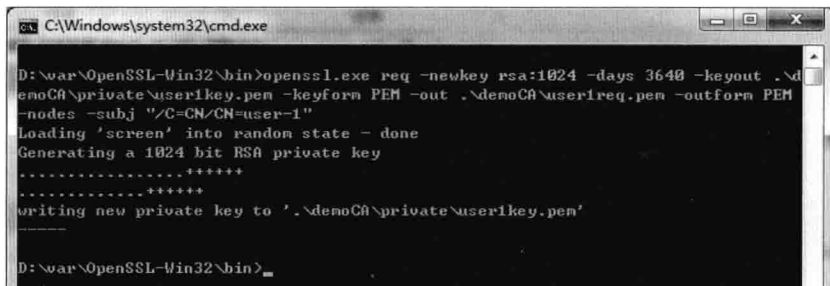


图 18-12 产生用户证书请求命令

接着，使用产生的用户请求文件，向刚才产生的 OpenSSL CA 证书申请生成用户证书。
执行命令：

```
openssl.exe ca -in .\demoCA\user1req.pem -out user1cert.pem -days 3640
```

openssl.exe ca 命令行参数可以通过执行命令“nssl.exe ca help”查看。

显示如图 18-13 所示的操作步骤。首先要求输入 CA 私钥的保护口令，这个值是在产

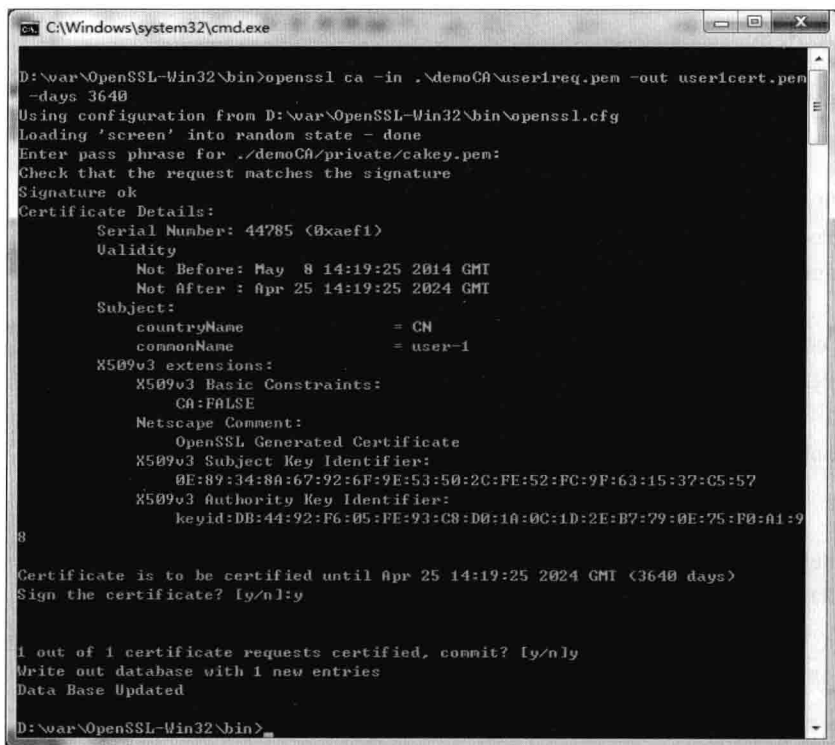


图 18-13 用户证书签发步骤

生 CA 证书时两次重复输入的保护口令，接着显示了证书信息，然后询问是否签发证书，输入 y 表示同意。然后显示是否提交证书请求，输入 y 表示提交。命令行显示将产生的证书提交到证书库成功。

产生的证书除了在当前目录存放一份名为 user1cert.pem 的证书文件外，还存放在.\demoCA\newcerts 目录下，以证书的序列号命名，如刚才产生的 user-1 证书的序列号为 AEF1，证书的文件名为 AEF1.pem。

打开产生的用户证书，可以看到图 18-14 所示信息。

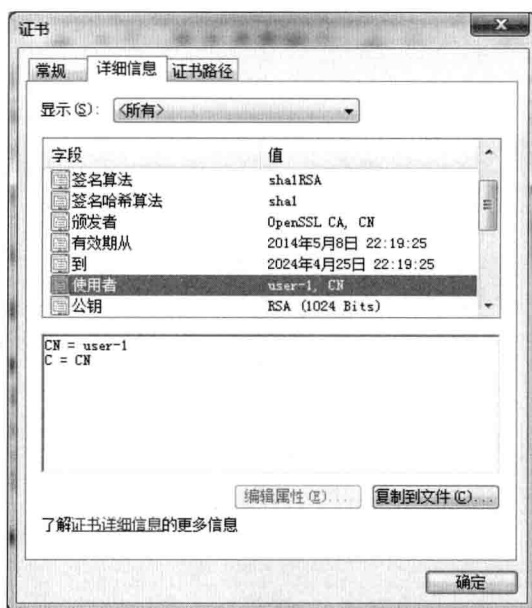


图 18-14 签发的 user-1 用户证书

重复同样的步骤，可以签发多张证书，此处签发 user-2、user-3 证书。

产生 user-2 证书，执行命令：

```
openssl.exe req -newkey rsa:1024 -days 360 -keyout .\demoCA\private\user2key.pem -keyform PEM
-out .\demoCA\user2req.pem -outform PEM -nodes -subj "/C=CN/CN=user-2"
openssl.exe ca -in .\demoCA\user2req.pem -out user2cert.pem -days 360
```

产生 user-3 证书，执行命令：

```
openssl.exe req -newkey rsa:1024 -days 360 -keyout .\demoCA\private\user3key.pem -keyform PEM
-out .\demoCA\user3req.pem -outform PEM -nodes -subj "/C=CN/CN=user-3"
openssl.exe ca -in .\demoCA\user3req.pem -out user3cert.pem -days 360
```

18.1.4 生成并下载 CRL

1. 生成空 CRL

执行命令“openssl.exe ca -gencrl -out .\demoCA\crl\democrl.crl”，生成空 CRL，如图 18-15 所示，需要输入 CA 私钥的保护口令。

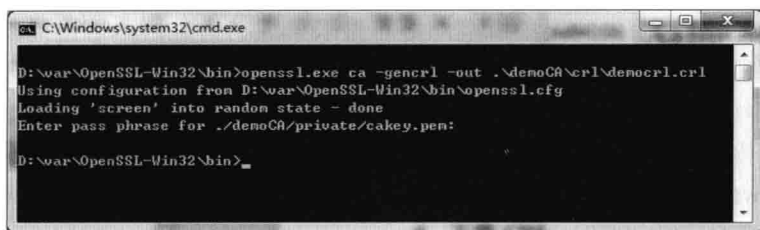


图 18-15 产生 CRL 文件

执行命令“openssl.exe crl -in .\demoCA\crl\democrl.crl -text -noout”，显示 CRL 内容，如图 18-16 所示。

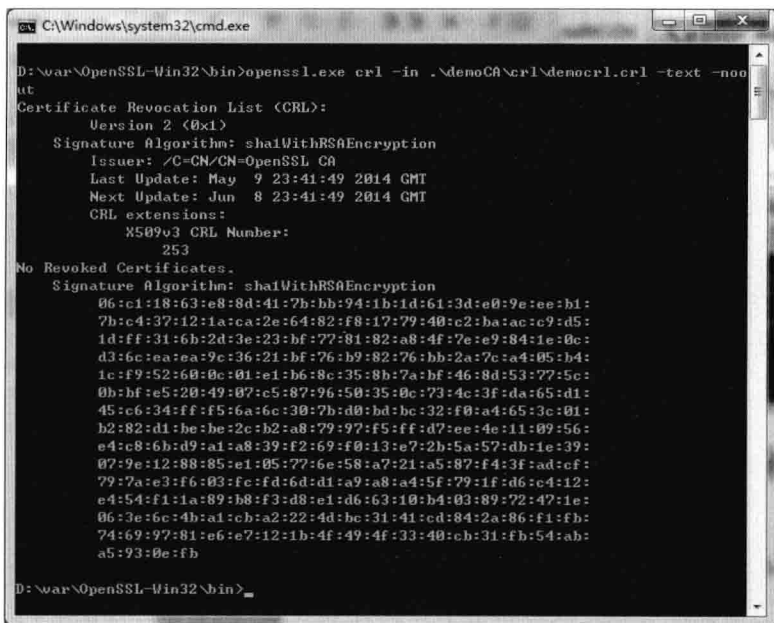


图 18-16 无吊销证书的 CRL 内容

2. 作废证书

执行命令“openssl.exe ca -revoke user3cert.pem”，作废证书 user3，如图 18-17 所示。

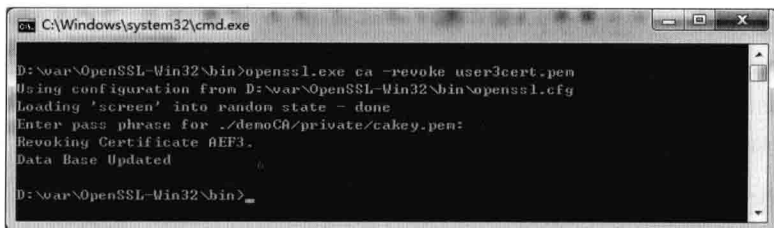


图 18-17 吊销 user-3 用户证书

3. 生成新 CRL

执行命令“openssl.exe ca -gencrl -out .\demoCA\crl\democrl.crl”，生成新 CRL，如图 18-18

所示。

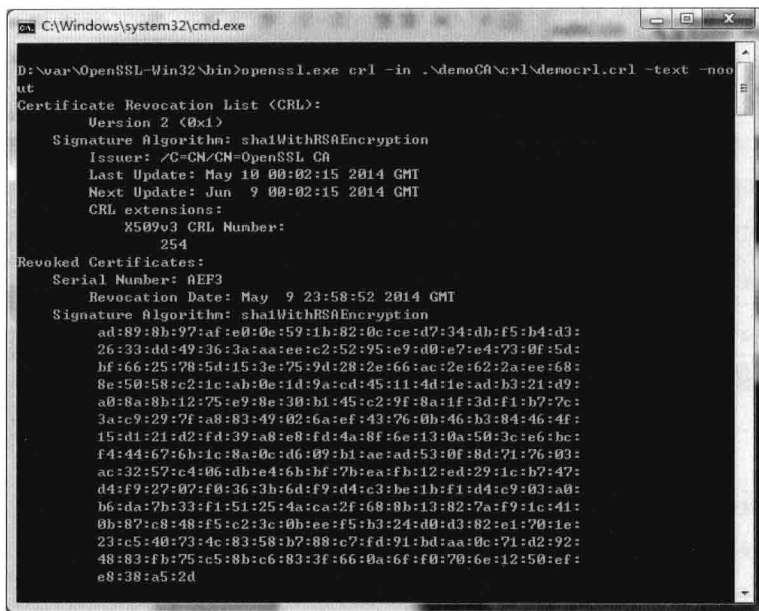


图 18-18 包含 user-3 用户证书的 CRL 内容

可以看到 user-3 用户对对应证书的序列号 AEF3 已经在 CRL 里了。在 Windows 环境下双击产生的 CRL 文件 democrl.crl，显示如图 18-19 所示信息。

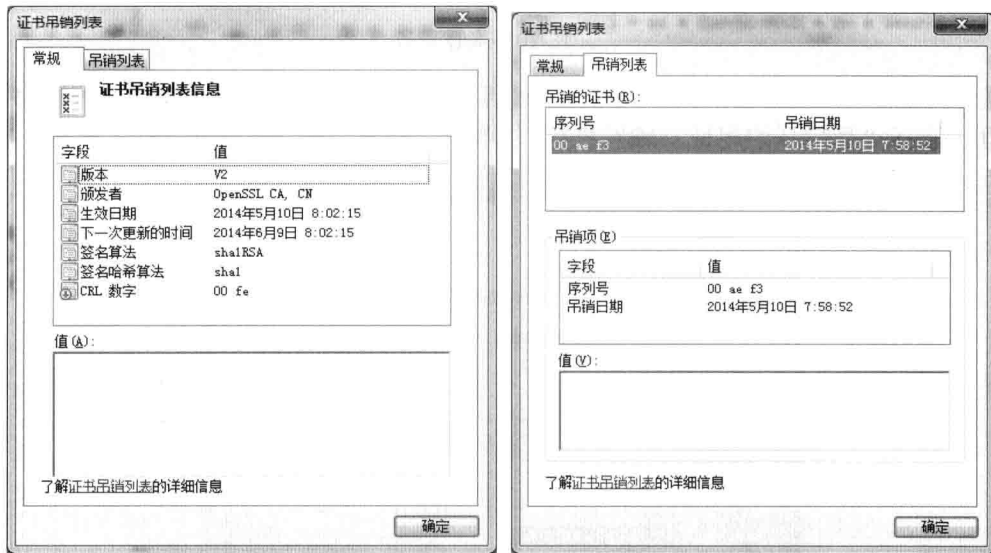


图 18-19 Windows 下 CRL 文件显示内容

可以看到 CRL 签发者为 OpenSSL CA，即前面签发的 CA 证书。作废证书序列号为 00 ae f3（证书序列号前面增加了 00），实际序列号就是 user-3 证书的 AEF3。

把 CRL 文件 democrl.crl 复制出来就可以使用了。

18.1.5 导入 CA 证书到 IE 可信任证书库

为确保浏览器信任 OpenSSL CA 签发的证书，需要将 OpenSSL CA 证书导入浏览器中的可信任根证书库中。

双击 CA 证书文件 cacert.pem，将出现证书信息界面。单击“安装证书”按钮，将出现证书导入向导界面，单击“下一步”按钮，如图 18-20 所示。

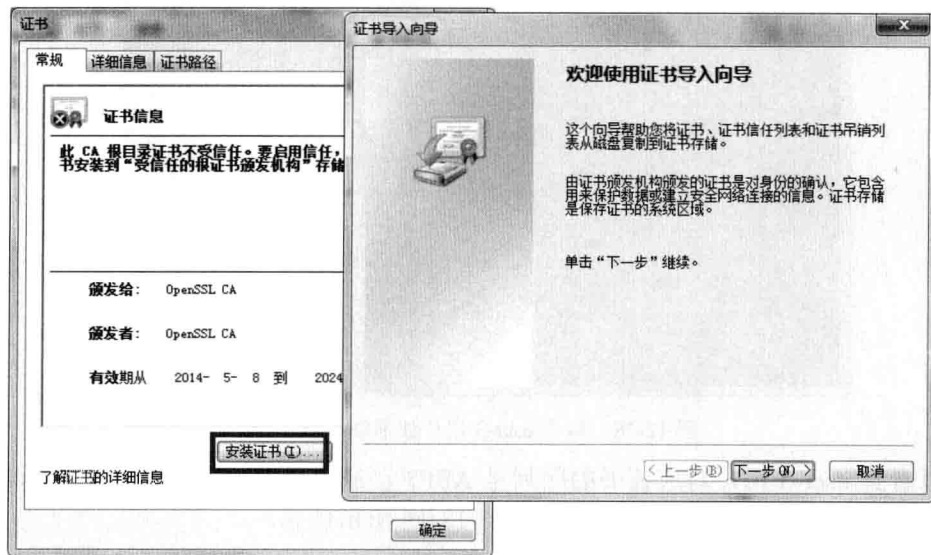


图 18-20 安装证书

选中“将所有的证书放入下列存储”，然后单击“浏览”按钮，将弹出“选择证书存储”对话框，选择“受信任的根证书颁发机构”，单击“确定”按钮，如图 18-21 所示。

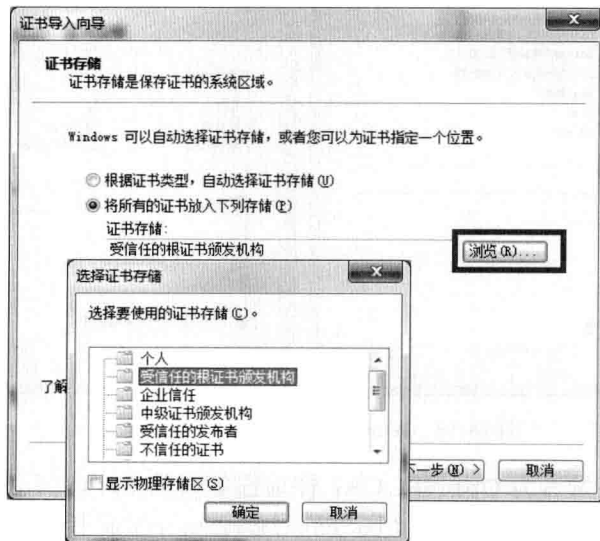


图 18-21 证书导入选择界面

单击“完成”按钮后，将弹出“安全性警告”对话框，系统提示“Windows 不能确认证书是否来自 OpenSSL CA”等警告信息，单击“是”按钮，弹出“导入成功”对话框，如图 18-22 所示。

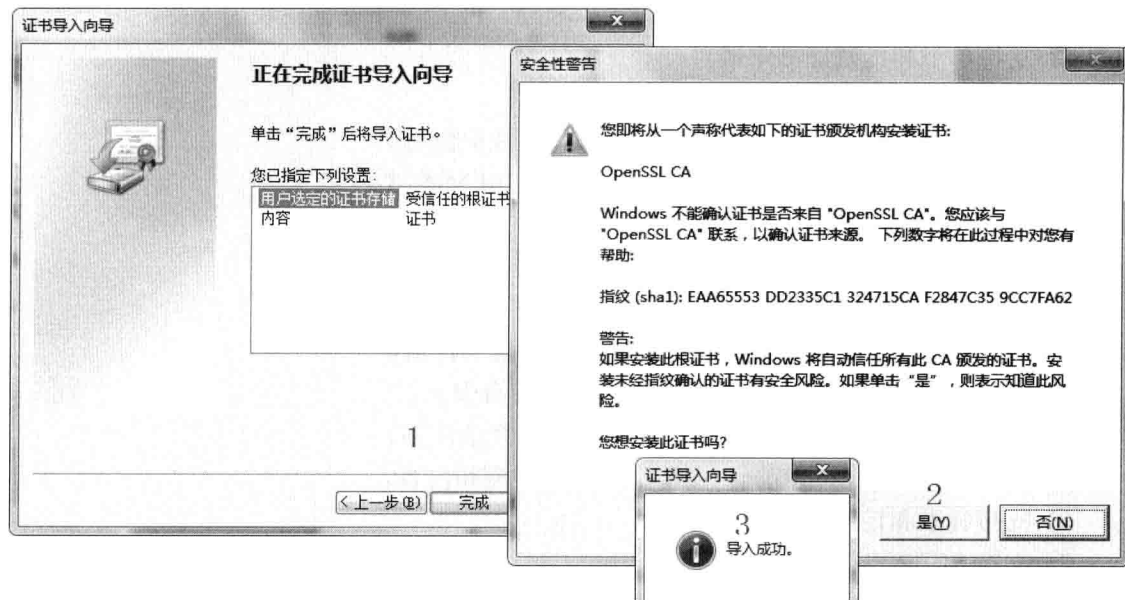


图 18-22 导入证书确认界面

18.2 EJBCA 示例

18.2.1 简介

EJBCA (Enterprise Java Bean Certificate Authority) 是一个全功能的 CA 系统软件，它基于 J2EE 技术，并提供了一个强大的、高性能的、基于组件的 CA。EJBCA 兼具灵活性和平台独立性，能够独立使用，也能和任何 J2EE 应用程序集成。EJBCA 有如下特点：

- ① 支持多个 CA 或多级 CA，可在一个 EJBCA 实例中建立多个 PKI 基础设施。
- ② 根 CA 和子 CA 数量不受限制。支持从其他 CA 申请交叉证书，从桥 CA 申请桥证书，给其他 CA 颁发交叉证书。
- ③ 从公开运营 CA 获取自己的 CA 证书。
- ④ 遵循 X509 和 PKIX 相关标准。
- ⑤ 支持 RSA 密钥长度到 8192 位。
- ⑥ 支持 DSA 密钥长度到 1024 位。
- ⑦ 支持 ECDSA 密钥算法。
- ⑧ 支持多种哈希算法签名，包括 SHA1、SHA2。

- ⑨ 兼容 NSA SUITE B 算法和证书。
- ⑩ 支持 X.509 证书和卡验证证书 (Card Verifiable certificates)。
- ⑪ 支持硬件安全模块, 内建对 Thales/nCipher、SafeNet Luna、SafeNet ProtectServer、Utlimaco CryptoServer、AEP Keyper、ARX CoSign、PKCS#11 安全模块的支持。
- ⑫ 支持单个和批量证书申请。
- ⑬ 颁发满足大多数服务器的 SSL/TLS 证书。
- ⑭ 支持通过插件方式扩展管理员注册和用户注册流程。
- ⑮ 服务器和客户端证书支持 PKCS12、JKS、PEM 格式。
- ⑯ 支持 Firefox、IE 等浏览器。
- ⑰ 通过 API 方式支持其他应用进行证书注册。
- ⑱ 注册 VPN 用户证书时, 可产生 OpenVPN 证书安装包。
- ⑲ 移动注册支持, 如以 SCEP 方式支持 iOS 证书注册。
- ⑳ 支持 CRL 生成, 及符合 RFC5280 的 CRL 分发点。
- ㉑ 支持 Windows、Linux、Mac OS X 智能卡登录证书。
- ㉒ 通过证书模板 (profiles) 支持不同证书类型和内容。
- ㉓ 支持标准和定制扩展项。
- ㉔ 支持 SCEP 协议。
- ㉕ 支持 RFC3739 (Qualified Certificate Statement), 能够发布 EU/ETSI 限定证书。
- ㉖ 支持 OCSP, 包括 AIA 扩展项。
- ㉗ 支持 RFC4387 通过 HTTP 获取 CA 证书和 CRL。
- ㉘ 支持德国通用 PKI SigG CertHash OCSP 扩展。
- ㉙ 支持 CMP (RFC4210 和 RFC4211)。
- ㉚ 支持异步 XKMS。
- ㉛ 支持密钥恢复。

EJBCA 系统支持的部署结构如图 18-23 所示。

在防火墙 1 后面是 RA 服务器和 OCSP 服务 (和 LDAP 查询服务器) 集群, 在 RA 和 CA 之间是防火墙 2, CA 服务器受防火墙 2 保护。为了数据安全性, 对 CA 数据库进行备份, 建立备份数据库服务器。在后续测试中, 为了方便, 把相关组件安装在一台应用服务器中。

18.2.2 安装配置

安装配置 EJBCA 相对复杂些, 需要分多个步骤依次进行:

- ① 下载安装 JDK。
- ② 下载安装 JDK 策略文件。
- ③ 下载安装 Apache Ant。
- ④ 下载 EJBCA 文件。

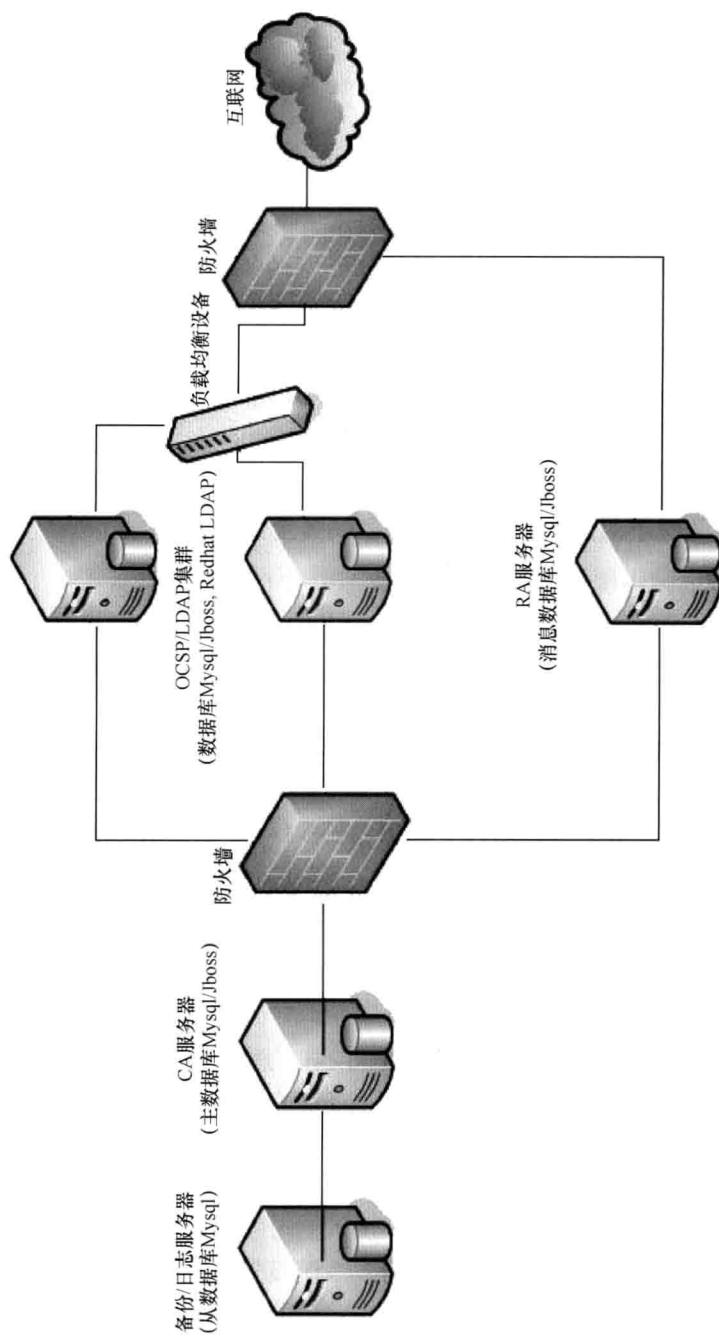


图 18-23 EJBCA 系统部署结构

- ⑤ 安装配置数据库。
- ⑥ 下载安装 JBoss AS 服务器。
- ⑦ 配置 JBoss AS 应用服务器。
- ⑧ 配置 EJBCA 参数文件。
- ⑨ 部署 EJBCA 到 JBoss 应用服务器。
- ⑩ 导入管理员证书。

1. 下载 JDK 1.7

使用 JDK1.7 最新版本，此处使用了 32 位版本。下载网址为：

<http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html>。

选择“Accept License Agreement”，然后单击 `jdk-7u55-windows-i586.exe` 链接进行下载。

实际下载地址为：

<http://download.oracle.com/otn-pub/java/jdk/7u55-b13/jdk-7u55-windows-i586.exe>。

下载完成后进行安装：把 JDK 安装在 `D:\Program Files\Java\jdk1.7` 目录下，设置环境变量 `JAVA_HOME=D:\Program Files\Java\jdk1.7`。

2. 下载 JDK 1.7 版本对应的 JCE 未限制密码强度策略文件

使用 JDK 1.7 对应的未限制密码强度策略文件，下载网址为：

<http://www.oracle.com/technetwork/java/javase/downloads/jce-7-download-432124.html>。

选择“Accept License Agreement”，然后下载 `UnlimitedJCEPolicyJDK7.zip` 文件。实际下载地址为：

<http://download.oracle.com/otn-pub/java/jce/7/UnlimitedJCEPolicyJDK7.zip>。

下载后需要替换 JAVA 安装目录下的安全策略文件：首先展开下载后的文件 `UnlimitedJCEPolicyJDK7.zip` 到临时目录，然后备份 `%JAVA_HOME%\jre\lib\security` 目录下的 `local_policy.jar` 和 `US_export_policy.jar`（例如在 `security` 目录下建立一个 `org` 子目录，把这两个文件复制到 `org` 目录下），最后把展开后同名称的两个文件复制到 `security` 目录下，覆盖原先的文件。

3. 下载 Apache 的 Ant 环境

下载网址为 <http://ant.apache.org/bindownload.cgi>，选择 `apache-ant-1.9.4-bin.zip`（实际下载网址为 <http://www.apache.org/dist/ant/binaries/apache-ant-1.9.4-bin.zip>）。

下载完成后，解压到 `D:\var\ejbca` 目录下，产生子目录 `apache-ant-1.9.4`。设置环境变量 `ANT_HOME=D:\var\ejbca\apache-ant-1.9.4`。

同时，把 `D:\var\ejbca\apache-ant-1.9.4\bin` 加入到可执行路径中：进入“控制面板”→“系统和安全”→“系统”，单击左边导航“高级系统设置”进入“系统属性”对话框；单击“环境变量”按钮，进入“环境变量”对话框；选择“PATH”变量，单击“编辑”按钮，进入编辑对话框，在“变量值”处增加“`D:\var\ejbca\apache-ant-1.9.4\bin;`”，其中分号（`;`）是分隔符，如图 18-24 所示。

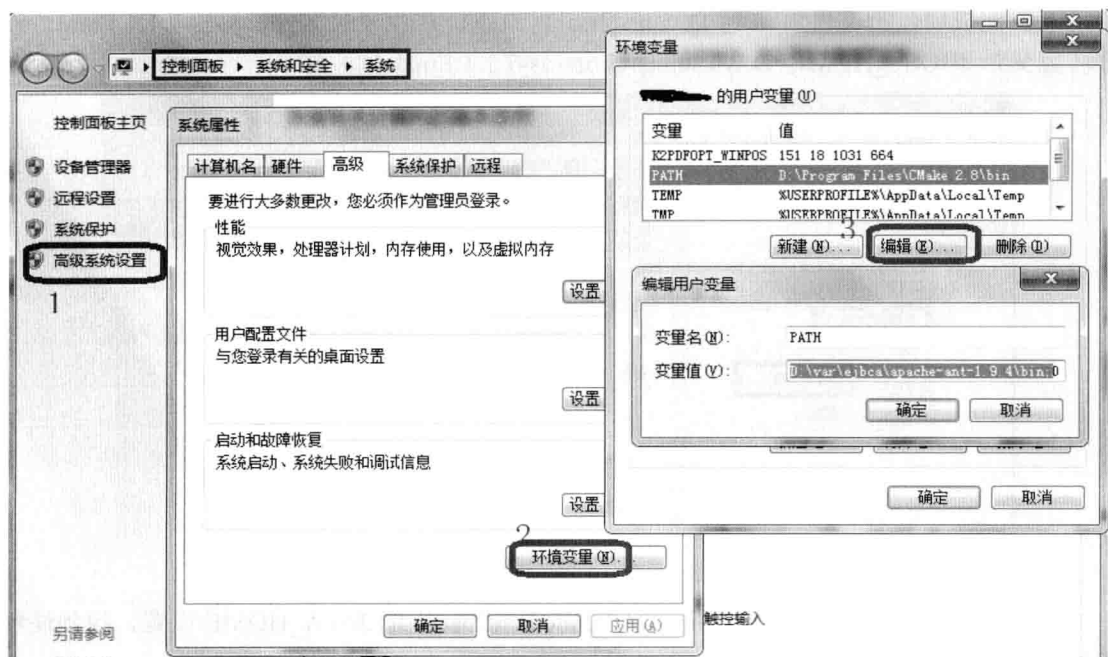


图 18-24 设置环境变量

如果没有“PATH”变量参数，可以单击“新建”按钮，增加一个，其他步骤同上。

4. 下载 EJBCA

下载网址为 <http://www.ejbc.org/download.html>，选择 EJBCA Community 6.1.1 版本。实际下载地址为：http://sourceforge.net/projects/ejbc/files/ejbc6/ejbc_6_1_1/ejbc_ce_6_1_1.zip。

下载完成后，解压到 D:\var\ejbca 目录下，产生子目录 ejbca_ce_6_1_1。设置环境变量 EJBCA_HOME=D:\var\ejbca\ejbca_ce_6_1_1。

5. 安装配置数据库

从 <http://www.enterprisedb.com/crossover-thankyou> 下载最新 Windows 稳定版 PostgreSQL 数据库安装程序（<http://get.enterprisedb.com/postgresql/postgresql-9.3.4-3-windows.exe>），下载完成后进行安装，基本采用默认设置即可，把口令设置为 postgres。

安装完成后，选择 JDBC 组件进行安装，安装后的 JDBC 文件为 postgresql-9.3-1100.jdbc4.jar。

为了能够顺利安装 EJBCA，需要预先创建 ejbca 数据库：使用 pgAdmin 管理工具，连接上数据库服务后，选中界面左边的“数据库”，选中鼠标右键菜单中的“新建数据库”，进入“新建数据库”对话框；输入名称“ejbca”，其他采用缺省值，单击“确定”按钮即可，如图 18-25 所示。

6. 安装 JBoss AS 服务器

下载网址为 <http://jbossas.jboss.org/downloads/>，选择 jboss-as-7.1.1.Final.zip。实际下载地址为 <http://download.jboss.org/jbossas/7.1/jboss-as-7.1.1.Final/jboss-as-7.1.1.Final.zip>。

下载完成后，把文件解压到 D:\var\ejbca 目录下，产生子目录 jboss-as-7.1.1.Final，设置环境变量 JBOSS_HOME=D:\var\ejbca\jboss-as-7.1.1.Final。

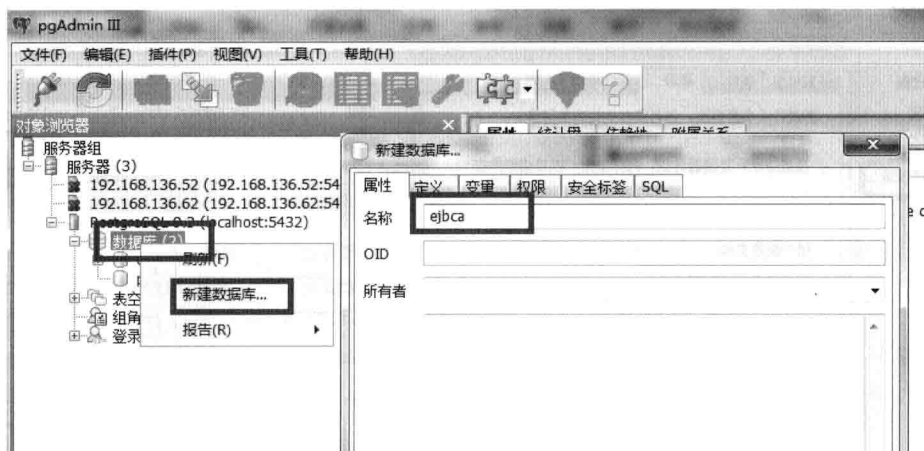


图 18-25 创建 ejbca 数据库

修改文件%JBOSS_HOME%\bin\standalone.conf.bat 中的 JAVA_HOME 设置，找到注释掉的 JAVA_HOME，在下面增加一行：

```
set "JAVA_HOME=D:\Program Files\Java\jdk1.7"
```

同样，编辑文件%JBOSS_HOME%\bin\jboss-cli.bat，在开始处，设置 JAVA_HOME 变量：

```
set "JAVA_HOME=D:\Program Files\Java\jdk1.7"
```

7. 配置 JBoss AS 应用服务器

(1) 安全模块配置

由于在 Windows 环境下，JBoss AS 和 Oracle 的 JDK 安全模块有兼容问题，需要重新设置。打开%JBOSS_HOME%\standalone/configuration 目录下的配置文件 standalone.xml，找到<subsystem xmlns="urn:jboss:domain:ee:1.0"/>标签，修改为：

```
<subsystem xmlns="urn:jboss:domain:ee:1.0">
  <ear-subdeployments-isolated>true</ear-subdeployments-isolated>
  <global-modules>
    <module name="org.bouncycastle" slot="main"/>
  </global-modules>
</subsystem>
```

建立%JBOSS_HOME%\modules/org/bouncycastle/main 目录，复制%EJBCA_HOME%\lib 下的文件 bcprov-jdk15on-149.jar 到%JBOSS_HOME%\modules/org/bouncycastle/main 目录下，并创建 module.xml 文件。module.xml 文件的内容为：

```
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.1" name="org.bouncycastle">
  <resources>
    <resource-root path="bcprov-jdk15on-149.jar"/>
  </resources>
</module>
```

```

</resources>
<dependencies>
    <module name="javax.api"/>
    <module name="org.jboss.logging"/>
    <module name="org.jboss.modules"/>
</dependencies>
</module>

```

(2) 增加对 SUN 类库模块的支持

为了使用 SUN 类库提供的 TCP 和 PKCS#11 功能, 需要增加 SUN 类库模块输出目录, 打开 %JBOSS_HOME%/modules/sun/jdk/main 目录下的 module.xml 文件, 在标签 module/dependencies/system/paths 路径下, 增加如下元素:

```

<path name="sun/security/action"/>
<path name="sun/security/x509"/>
<path name="sun/security/pkcs11"/>
<path name="sun/security/pkcs11/wrapper"/>

```

(3) 设置 PostgreSQL JDBC 驱动

建立 %JBOSS_HOME%/modules/org/postgresql/main/ 目录, 在此目录下复制一份 postgresql-9.3-1100.jdbc4.jar 驱动文件, 并创建 module.xml 文件。module.xml 文件内容为:

```

<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.0" name="org.postgresql">
    <resources>
        <resource-root path="postgresql-9.3-1100.jdbc4.jar"/>
    </resources>
    <dependencies>
        <module name="javax.api"/>
        <module name="javax.transaction.api"/>
    </dependencies>
</module>

```

(4) 通过命令行进行其他配置

首先启动 JBoss 应用服务, 打开命令行窗口, 执行命令 %JBOSS_HOME%/ bin/ standalone.bat, 等待服务器启动完成。

打开新的命令行窗口, 执行命令 %JBOSS_HOME%/bin/jboss-cli.bat。输入 connect 指令, 如图 18-26 所示。

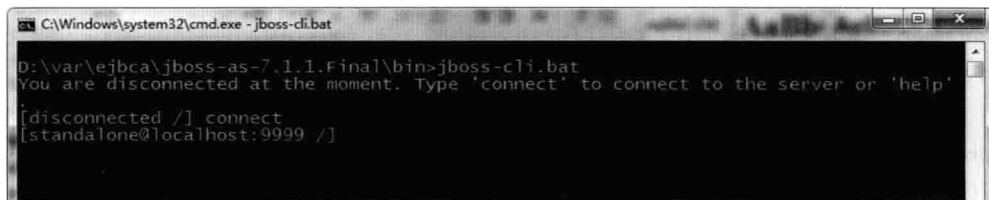


图 18-26 JBoss 命令行窗口

增加 PostgreSQL 数据源，执行命令：

```
/subsystem=datasources/jdbc-driver=org.postgresql.Driver:add(driver-name=org.postgresql.Driver,driver-module-name=org.postgresql,driver-xa-datasource-class-name=org.postgresql.xa.PGXADatasource)
:reload
```

修复 WSDL 访问位置生成错误，执行命令：

```
/subsystem=webservices:write-attribute(name=wsdl-host, value=jbossws.undefined.host)
:reload
```

如果 HornetQ 启动有问题，使用 JAVA NIO 处理，执行命令：

```
/subsystem=messaging/hornetq-server=default:write-attribute(name=journal-type,value=NIO)
:reload
```

打开日志跟踪，执行命令：

```
/system-property=org.jboss.as.logging.per-deployment:add(value=false)
/subsystem=logging/logger=org.ejbca:add
/subsystem=logging/logger=org.ejbca:write-attribute(name=level, value=DEBUG)
/subsystem=logging/logger=org.cesecore:add
/subsystem=logging/logger=org.cesecore:write-attribute(name=level, value=DEBUG)
```

8. 配置 EJBCA 参数文件

在安装 EJBCA 服务前，需要进行相关参数配置，主要配置文件有 5 个：ejbca.properties、install.properties、cesecore.properties、web.properties、database.properties。

从%EJBCA_HOME%\conf 路径下，分别复制 5 个文件 ejbca.properties.sample、install.properties.sample、cesecore.properties.sample、web.properties.sample、database.properties.sample，并改名为 ejbca.properties、install.properties、cesecore.properties、web.properties、database.properties。

配置文件中对每个配置项都进行了说明，基本上采用缺省配置即可。

(1) 配置 ejbca.properties

首先设置应用服务器路径，配置 appserver.home 到%JBoss_HOME%目录。为了测试方便，打开提示信息，修改 ejbca.productionmode 的值为 false。缺省情况下，禁止通过系统文件修改 EJBCA 的缺省配置。基本 CA 参数配置中，设置密钥库口令，此处采用缺省值“foo123”，其他参数采用缺省值。

(2) 配置 install.properties

把 CA 的 DN 名改为 CN=ManagementCA,O=EJBCA,C=CN，其他采用缺省值。

(3) 配置 cesecore.properties

设置密钥库口令，此处采用缺省值“foo123”。设置证书最大终止日期 ca.toolateexpireddate 为 2038-01-19 03:14:08+00:00，为缺省值。设置缺省语言为英语，intresources.Preferred language=EN。其他采用缺省值。

(4) 配置 web.properties

把 java.trustpassword 值设为 trustpass，把 superadmin.batch 改为 false，把 httpserver.password 的值改为 serverpass，把 httpserver.dn 的值改为 CN=\${httpserver.hostname}, O=EJBCA,C=CN，其他保持不变。

(5) 配置 database.properties

使用 postgresql 数据库，打开 database.properties 文件，依次设置为：

```
database.name=ejbca
database.url=jdbc:postgresql://127.0.0.1/ejbca
database.driver=org.postgresql.Driver
database.username=postgres
database.password=postgres
database.useSeparateCertificateTable=true
```

9. 部署 EJBCA 到 JBoss 应用服务器

启动 JBoss 后，打开命令行窗口，进入 %EJBCA_HOME% 目录，执行命令 “ant deploy”，此命令执行编译和部署 EJBCA 到 JBoss 中。命令执行完成后，系统提示成功。信息如下：

```
BUILD SUCCESSFUL
Total time: 2 minutes 16 seconds
```

执行命令 “ant install”，此命令完成 CA 初始化，包括产生所有的证书、密钥，部署数据源、服务，配置 servlet 容器使用的密钥库和信任库文件，最后生成的管理员密钥在 %EJBCA_HOME%/p12 目录下。

“ant install” 命令只能执行一次，它在数据库中产生了大量信息，再次执行会出错。如果需要重新执行 CA 初始化，在执行 “ant install” 命令前要删除 ejbca 数据库中的所有表。

10. 导入管理员证书

部署完成后，需要把管理员证书导入到浏览器中。在 %EJBCA_HOME%/p12 目录下有管理员的 PKCS#12 格式证书密钥文件 superadmin.p12，需要导入到浏览器中。双击 superadmin.p12 文件后，出现 “证书导入向导” 界面，按照界面提示进行操作即可。其中，在 “密码” 对话框中，输入私钥保护密码为 ejbca（私钥保护密码为 web.properties 中设置的 superadmin.password 的值，在配置文件中已设置为 ejbca），如图 18-27 所示。

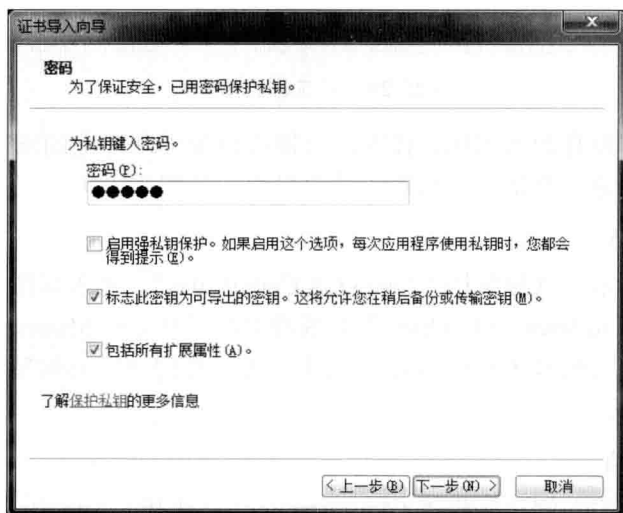


图 18-27 PKCS#12 文件私钥保护密码

11. 修改配置文件后重新部署

如果配置文件有改变,不需要重新安装 EJBCA,只需要执行命令“ant clean deployear”即可完成部署。

18.2.3 申请证书

配置完成后,需要重新启动 JBoss 应用服务器:通过 CTRL+C 终止服务后,重新在命令行窗口执行命令%JBOSSE_HOME%\bin\standalone.bat 启动服务。

1. 管理员登录

打开 IE 浏览器,在地址栏输入 https://localhost:8443/ejbca/, 出现图 18-28 所示的证书选择界面,选择 SuperAdmin 并单击“确定”按钮。



图 18-28 证书登录选择框

由于根 CA 证书没有加入本地信任库,可能会出现“此网址的安全证书有问题”警告窗口,单击“继续浏览此网站”后可进入管理界面,见图 18-29。

2. 导出 CA 证书

在管理界面中单击左侧导航栏“Fetch CA Certificates”,进入如图 18-30 所示界面。

选择“Download to Internet Explorer”后获得 CA 证书文件 ManagementCA.crt。双击该文件,可将该证书导入到本地 Windows 证书库“受信任的根证书颁发机构”中。具体操作步骤请参见 18.1.5 节。

3. 提交申请资料

在管理界面中单击左侧导航栏“Administration”,选择“Add End Entity”进入 EE 注册界面,如图 18-31 所示。

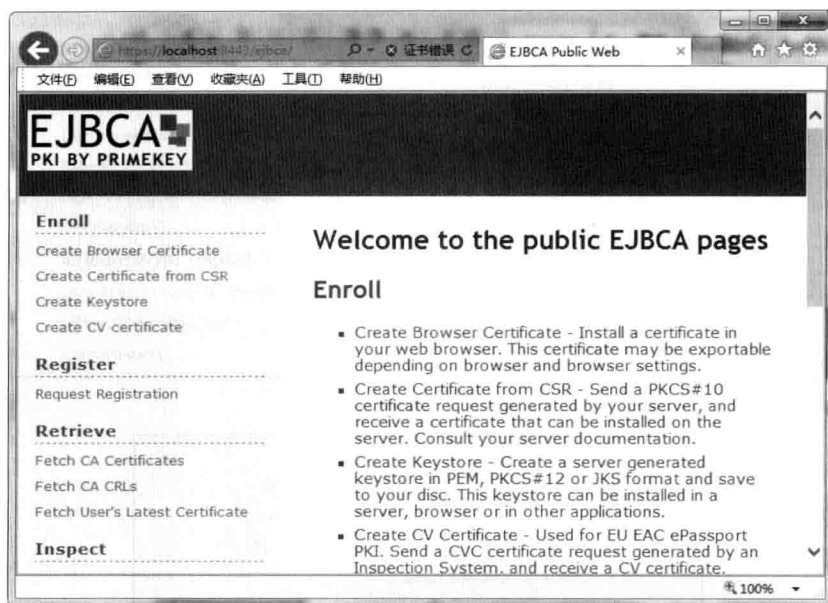


图 18-29 EJBCA 管理界面



图 18-30 导出 CA 证书

为了演示方便，只添加必要的注册信息，包括用户名（ejbcauser-1）、口令（123456）、CN（user-1）、Token（选择 P12 文件）。单击“Add”按钮，提示注册成功，并在底部列出已注册用户，如图 18-32 和图 18-33 所示。

4. 下载证书

在管理界面中单击左侧导航栏“Create Browser Certificate”，输入注册时的用户名和密码（分别是 ejbcauser-1 和 123456），如图 18-34 所示。

图 18-31 证书申请界面

图 18-32 用户注册

Username	CN	OU	O (organization)	
ejbcauser-1	user-1			View End Entity Edit End Entity

图 18-33 用户注册成功提示

图 18-34 输入注册用户名和口令

单击“OK”按钮后,将出现密钥产生界面,如图 18-35 所示。选择密钥长度 Key length 为 1024 位,单击“Enroll”按钮,出现下载文件提示框,单击“确认”按钮后,等待文件下载完成。下载的文件是 P12 格式,包括证书和私钥,文件保护口令是注册时输入的 123456。

双击下载的 P12 文件,可以将该证书导入到本地 Windows 证书密钥库,与导入管理员证书类似。



EJBCA Token Certificate Enrollment

Welcome to keystore enrollment.

If you want to, you can manually install the CA certificate(s) in your browser, otherwise this will be done automatically when the certificate is retrieved.

Install CA certificates:

[Certificate chain](#)

Please choose a key length, then click OK to fetch your certificate.

Options

Leave values as default if unsure.

Key length: 1024 bits ▼

Certificate profile: ENDUSER ▼

图 18-35 选择密钥长度

对于客户端产生的 PKCS#10 格式的证书请求，可以通过管理界面中左侧导航栏“Create Certificate from CSR”功能，把 P10 证书请求粘贴到编辑框中，从而实现证书签发，如图 18-36 所示。



A PEM-formatted request is a BASE64 encoded certificate request starting with
-----BEGIN CERTIFICATE REQUEST-----
and ending with
-----END CERTIFICATE REQUEST-----

Enroll

Username

Enrollment code

Request file 未选择文件

or pasted request

Result type PEM - certificate only ▼

图 18-36 PKCS#10 格式的证书请求

18.2.4 下载 CRL

在管理界面中单击左侧导航栏“Fetch CA CRLs”，进入 CRL 下载页面，如图 18-37 所示。

选择左侧链接“DER format”（完整 CRL，不是 Delta CRL），下载完整 CRL 为 ManagementCA.crl 文件。双击该文件可查看其中的详细内容，见图 18-38。可以看到，其中作废的证书列表为空。

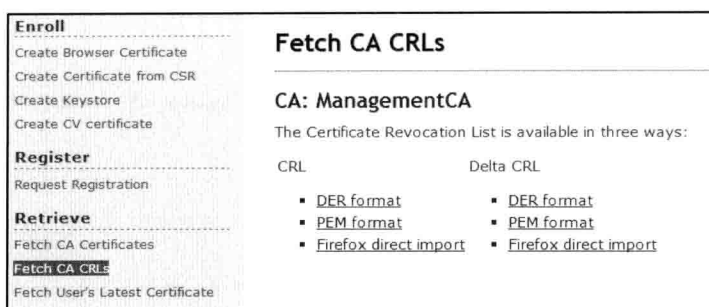


图 18-37 获取 CRL 界面

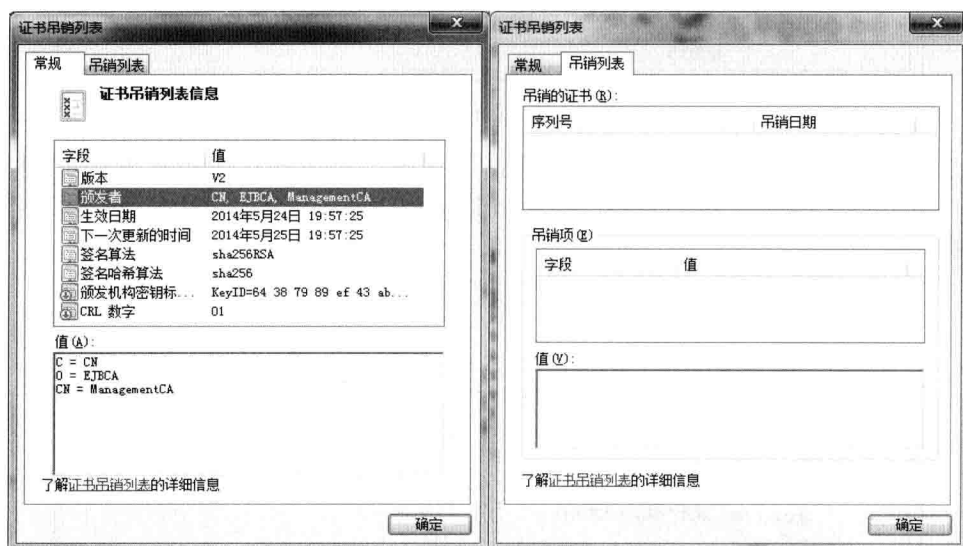


图 18-38 下载的 CRL 文件内容

为了展示非空 CRL，作废刚才产生的 ejbcauser-1 证书，在管理界面中单击左侧导航栏“Search End Entities”，出现查询界面，如图 18-39 所示。

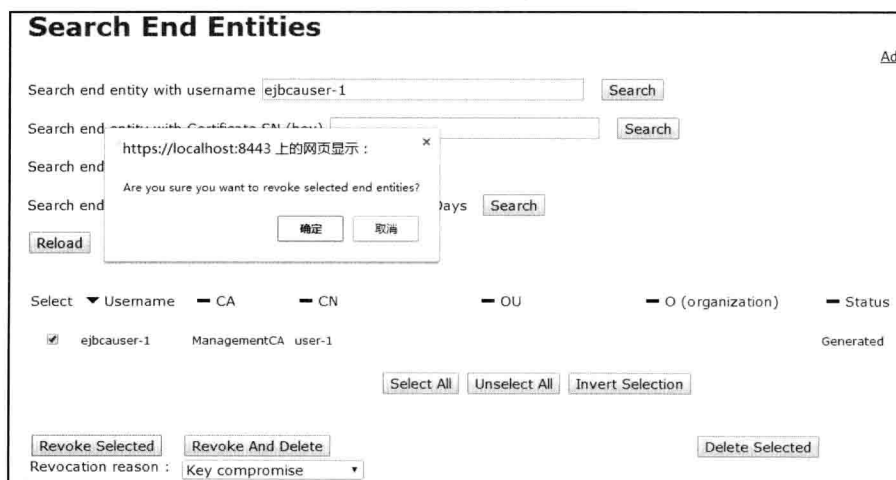


图 18-39 作废用户证书

在 Search end entity with username 中输入“ejbcauser-1”后，单击“Search”按钮进行查询。页面下边显示查询结果列表，勾选 ejbcauser-1 条目，在作废原因中选择“Key compromise”，单击“Revoke Selected”按钮，完成证书作废。

作废用户证书后，需要主动触发产生 CRL 文件。在管理界面中单击左侧导航栏“CA Structure & CRL”，出现证书作废页面，如图 18-40 所示。单击“Create CRL”按钮，完成 CRL 生成。

再次下载 CRL 后，可以看到有一条被作废的用户证书（ejbcauser-1 用户证书），如图 18-41 所示。

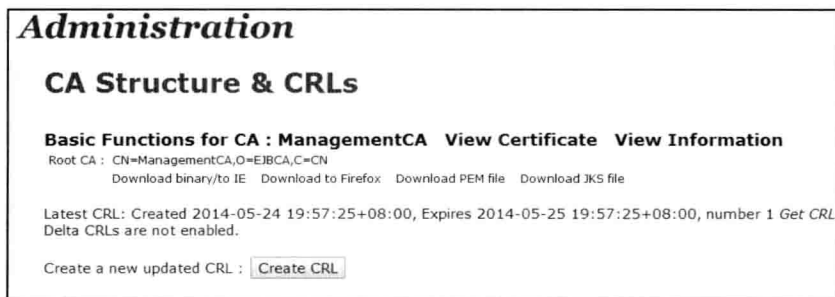


图 18-40 产生 CRL

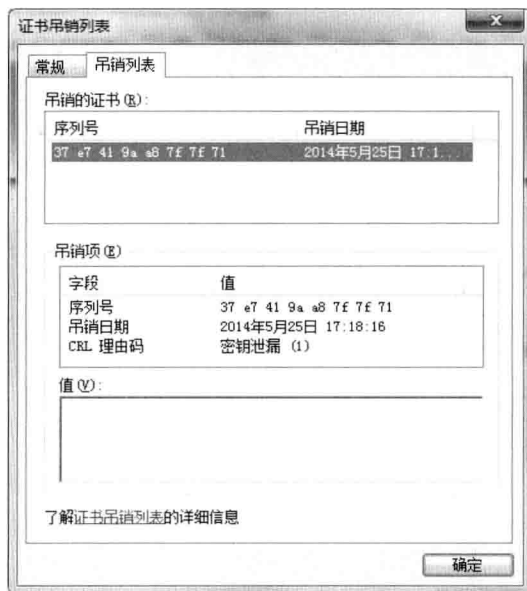


图 18-41 包含 ejbcauser-1 吊销证书的 CRL

除了手动触发生成 CRL 外，EJBCA 同时支持定时产生 CRL，即在某个设定时间产生 CRL 文件。

第五部分

PKI 之应用：使用网络身份证

第19章 基本应用

19.1 身份认证

身份认证服务提供一种鉴别实体真伪的方法。由于数字证书在申请时身份已经得到发证机构确认，所以把数字证书用于应用系统对用户的身分认证是非常合适的。在应用中集成数字证书进行身份认证时，主要是验证用户是否拥有证书对应的私钥，即验证用户私钥对一段特定数据的签名是否正确。

例如，网上银行系统中，用户使用 U 盾（即 USBKey）登录网银，U 盾中存储了用户的证书和私钥，并通过密码对 U 盾进行保护。网上银行证书登录界面如图 19-1 所示。

图 19-1 证书登录输入界面

界面输入元素含义分别为：

- ① 设备类型，表明采用的设备类型，网银支持多种类型的设备，如 U 盾、动态口令等；
- ② 用户 ID，一个容易记的名称；
- ③ 数字证书，显示设备中证书对应的使用者名称；
- ④ 证书密码，设备保护密码；
- ⑤ 验证码，图片对应字符，防止重复提交。

当用户单击“提交”按钮时，登录功能主要做了两件事：

① 通过“证书密码”输入值验证用户是否有访问 U 盾的权限，此密码保护 U 盾不被非法访问；

② 在获取访问 U 盾的权限后，使用 U 盾中的私钥对服务端传递来的一段随机数据进行签名，然后把签名值和证书提交到服务端，服务端使用发送给客户端的随机数据，以及用户提交的证书对签名值进行验证。如果验证通过，表明用户拥有证书对应私钥及其使用权限；如果验证没有通过，表明用户没有证书对应私钥的使用权限。

即使用户通过了签名值验证阶段，也不一定能够通过身份认证。一般情况下，业务系

统还会对用户证书的有效性进行验证。有效性验证包括：证书是否过期、是否作废、是否为指定 CA 签发、密钥用途等。

网上银行用户的身份认证（证书登录）过程是典型的挑战应答模式，由服务端给出挑战值，即一段随机数据，客户端对挑战值进行应答，即对随机数据进行签名，然后提交应答（包括签名值、证书等）。

出于安全考虑，为了防止重复攻击，每次身份认证时，服务端都应该给出不同的挑战值，使挑战值尽可能随机化。

《卫生系统数字证书应用集成规范》中给出了数字证书登录的认证流程，如图 19-2 所示。

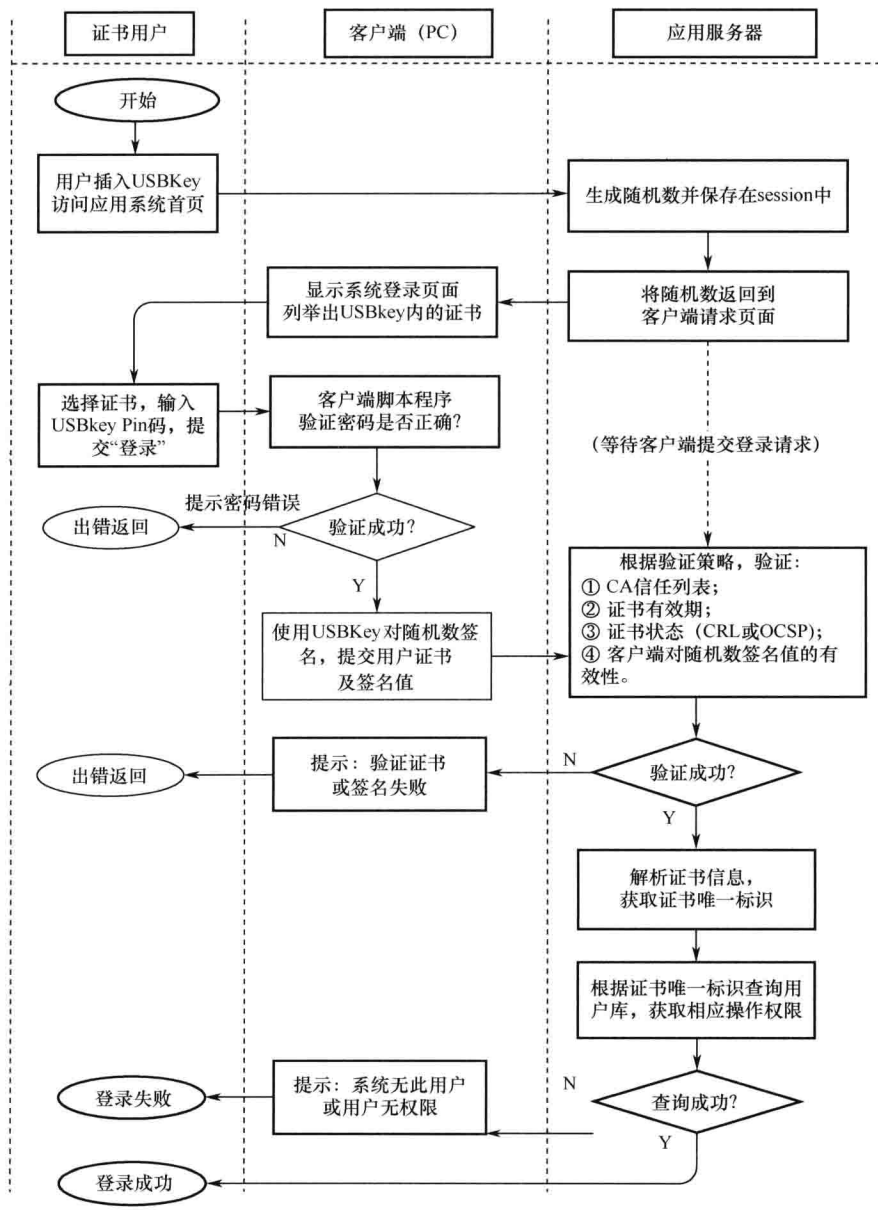


图 19-2 证书登录认证流程示意图

19.2 保密性

保密性是对数据进行加密的操作，使数据只能被授权的实体看到，它适用于以下几种场合。

(1) 自己加密，自己解密

在这种情况下，实体出于保护自己数据的目的，使用自己的证书公钥对数据进行加密，也只有自己的私钥才能解开。

例如，用户有敏感信息需要保存，这些信息不希望被其他人看到，这样他就可以用自己的证书公钥对敏感信息加密，因私钥由自己掌控，其他人即使拿到加密后的数据，也无法打开。

(2) 自己加密，他人解密

在这种情况下，用户出于保护要传递数据的目的，使用对方的证书公钥对数据进行加密，也只有对方的私钥才能解开。

例如，甲要向乙发送数据，而甲不希望除乙之外的其他人看到发送数据的内容，这时，甲就可以用乙的证书公钥加密数据，也只有乙能够用自己的私钥解开接收到的数据。平时所用的加密邮件，就属于此种类型应用。

由于使用公钥加密的效率较低，当需要加密的数据较多时，这种效率是不能接受的。为了解决这个问题，一般采用数字信封机制，即先产生一个对称密钥，使用对称密钥对数据进行加密，然后使用证书公钥对对称密钥加密，这样加密的对称密钥和加密后的数据形成了一个数字信封。解密时，进行反向操作，先用私钥解密对称密钥，然后用对称密钥解密加密的数据，得到数据原文。

构造数字信封时，一般使用 PKCS#7 中定义的数字信封格式，具体可参见第 5 章的内容。

下面以加密邮件说明数据的保密性。一封简单加密邮件包含如下信息：

```
Content-Type: application/pkcs7-mime; smime-type=enveloped-data;
name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m
rfvbnj756tbBghyHhHUujhJhjH77n8HHGT9HG4VQpfyF467GhIGfHfYT6
7n8HHGghyHhHUujhJh4VQpfyF467GhIGfHfYGT6jH7756tbB9H
f8HHGT6jH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
0GhIGfHfQbnj756YT64V
```

邮件内容类型为 application/pkcs7-mime，smime-type=enveloped-data 表明是信封数据，加密后邮件正文采用 Base64 编码。

发送加密邮件时，加密操作在发送邮件时执行，捕获电子邮件未加密原文，并使用预期收件人所特有的信息（证书公钥）对邮件进行加密。加密的邮件替换了原始邮件，然后将邮件发送至收件人。图 19-3 显示了电子邮件加密过程。

由于此操作需要收件人的唯一信息（证书公钥），因此邮件加密提供了保密性。只有预期收件人具有执行解密操作所需的信息，从而确保了只有预期的收件人能够查看邮件。

当收件人打开加密邮件时，会对加密邮件执行解密操作。此时，将同时检索加密的邮件和收件人的唯一信息。然后，使用收件人的唯一信息对加密邮件执行解密操作。此操作



图 19-3 加密邮件过程

返回未加密的邮件，然后该邮件将显示给收件人。如果邮件在传送过程中发生过改变，解密操作将失败。图 19-4 显示了解密电子邮件过程的顺序。



图 19-4 解密电子邮件

19.3 完整性

完整性是指在传输、存储信息或数据的过程中，确保信息或数据不被未经授权地篡改或在篡改后能够被迅速发现。这种方法实质是一种数据摘要过程，首先利用哈希函数计算数据哈希值，然后将数据及其哈希值一起发送到对方。对方收到数据后，重新利用哈希函数计算数据哈希值并与接收到的数据哈希值比对，进而判断数据是否被篡改。

由于哈希值在传输中可能被截取修改，需要对哈希值进行保护。一般通过数字签名的方式对哈希值进行保护，接收方对收到的数据的数字签名进行验证，如果数据原文被修改，则签名验证就不会通过。

在应用中，一般使用 PKCS#7 的签名数据结构组装数据和签名值。例如，发送签名邮件时，就是使用 PKCS#7 的签名数据结构封装消息，具体可参见第 5 章的内容。

下面以签名邮件说明数据的完整性。一封简单签名邮件包含如下信息。

```

Content-Type: application/pkcs7-mime; smime-type=signed-data;
name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m
  
```

```

567GhIGfHfYT6ghyHhHUujpfyF4f8HHGTrfvhJhjH776tbB9HG4VQbnj7
77n8HHGT9HG4VQpfyF467GhIGfHfYT6rfvbnj756tbBghyHhHUujhJhjH
HUujhJh4VQpfyF467GhIGfHfYGTrfvbnjT6jH7756tbB9H7n8HHGghyHh
6YT64V0GhIGfHfQbnj75
  
```

内容类型为 application/pkcs7-mime，smime-type=signed-data 表明是签名数据，签名后邮件正文采用 Base64 编码。

发送签名邮件时，执行签名操作所需要的信息只能由发件人提供。在签名操作中，通过捕获电子邮件并对邮件执行签名操作来使用此信息。此操作产生实际的数字签名。然后，此签名将附加到电子邮件中，并随同邮件一起发送。图 19-5 显示了邮件签名过程的顺序。



图 19-5 发送签名邮件

由于此操作需要来自发件人的唯一信息（私钥），因此数字签名提供了身份验证和数据完整性功能。此唯一信息可以证明邮件只能来自该发件人。

当收件人打开经过数字签名的电子邮件时，系统会对数字签名执行验证过程，并会从邮件中检索邮件所包含的数字签名。还会检索原始邮件，然后执行签名验证操作，如果签名验证通过，则证明邮件确实来自所声称的那个发件人；如果签名验证不通过，则将邮件标记为无效。图 19-6 显示了邮件验证过程的顺序。



图 19-6 验证签名邮件

同时采用数字签名过程和数字签名验证过程可验证电子邮件发件人的身份，并确定已签名的邮件中数据的完整性。

19.4 抗抵赖性

抗抵赖提供一种防止实体对其行为进行抵赖的机制，它从技术上保证实体对其行为的认可。由于实体的各种行为只能发生在它被信任之后，所以可通过时间戳标记和数字签名来审计实体的各种行为。通过将实体的各种行为与时间和数字签名绑定在一起使实体无法抵赖其行为。

采用单纯的数字签名方式不能起到抗抵赖作用，因为签名说明签名者对数据或行为进行了确认，但不能说明他什么时间进行了确认，因此还必须进行可信的第三方进行事件发生时间的证明，即需要用到时间戳服务。

简单地说，时间戳服务是对一段数据进行时间标记，并对时间标记进行签名。在 RFC 3161 中对时间戳协议进行了说明。可信时间戳服务的本质是将用户的电子数据的哈希值和权威时间源绑定，在此基础上通过时间戳服务中心（TSA）进行数字签名，产生不可伪造的时间戳数据。时间戳协议在第 20 章有详细说明。

为了达到抗抵赖效果，对原始数据的签名包括两部分，一是实体对数据的签名，二是时间戳服务中心对数据的签名。这两部分签名分别使用了 PKCS#7 和 CMS 的签名数据结

构,实际使用中,还需要把这两部分数据组合成一个数据,使数据的管理和使用更方便。根据 RFC3161 规范建议,一般把时间戳数据作为 PKCS#7 的一个属性,即 PKCS#7 签名数据中包含时间戳。

在实际应用中,经常使用签名和时间戳对要发布的软件进行签名,这样用户安装软件时,就可以对软件的可信性进行验证。一般使用专用签名工具完成带时间戳的签名。

例如,打开一个带时间戳签名的软件文件的属性界面,如图 19-7 所示。

单击“详细信息按钮”,显示如图 19-8 所示的签名和时间戳。可以看出签名时间为 2013 年 12 月 26 日 14 时 50 分 28 秒。



图 19-7 带时间戳的软件数字签名信息

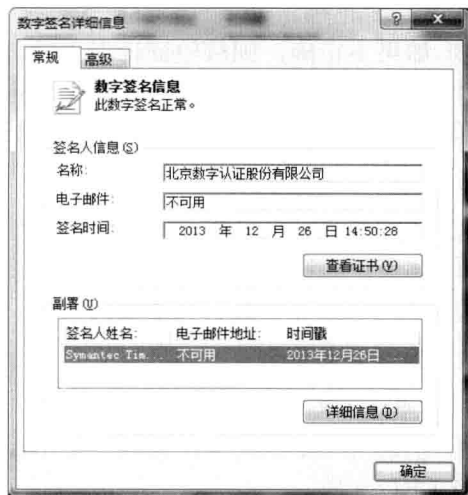


图 19-8 签名和时间戳

单击“详细信息”按钮,可以看到时间戳签发机构 (Symantec Time Stamping Services Signer - G4), 如图 19-9 所示。其证书信息如图 19-10 所示。



图 19-9 时间戳签发机构

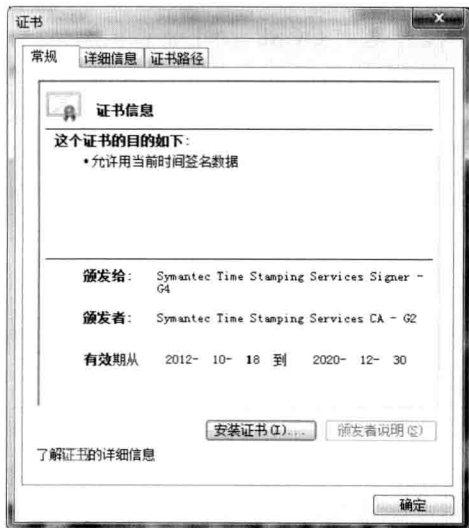


图 19-10 时间戳签发机构证书信息

19.5 证书有效性验证

用户在使用数字证书进行加密和验证数字签名时，必须首先验证证书的有效性。验证证书有效性主要包括以下步骤。

1. 验证 CA 签名

- ① 将待验证证书设置为当前证书。
- ② 根据当前证书中签发者信息查询签发者证书，并使用签发者证书验证当前证书中的数字签名是否正确。
- ③ 如果不正确，则待验证证书无效。
- ④ 如果正确且签发者证书是预先确定的可信任 CA 证书，则说明待验证证书的 CA 签名正确；否则，将签发者证书设置为当前证书，重复步骤②～④。

2. 验证证书有效期

- ① 如果待验证证书中 `notBefore` 在当前日期之后，则该证书未生效。
- ② 如果待验证证书中 `notAfter` 在当前日期之前，则该证书已过期。
- ③ 否则，该证书处于有效期内。

3. 验证证书状态

(1) 基于 CRL 验证证书状态

- ① 根据待验证证书中扩展项 `CRLDistributionPoints` 信息获得 CRL 地址，并根据该地址下载 CRL 文件。
- ② 检查 CRL 文件是否在有效期内，若不在则重新下载。
- ③ 验证 CRL 中数字签名是否正确，若不正确则重新下载。
- ④ 如果 CRL 中包括待验证证书的序列号，则说明该证书状态为无效；否则说明该证书状态为有效。

(2) 基于 OCSP 验证证书状态

- ① 将待验证证书的序列号组织成 OCSP 请求包。
- ② 将 OCSP 请求包发送给 OCSP 服务器。
- ③ 从 OCSP 服务器获得 OCSP 响应包。
- ④ 解析 OCSP 响应包获得该证书的当前状态。

4. 验证证书其他属性

- ① 验证证书密钥用途：`KeyUsage`、`ExtKeyUsage`、`NetscapeCertType`。
- ② 验证证书策略：`CertificatePolicies`。
- ③ 其他专用属性。

第 20 章 通用应用技术

20.1 SSL/TLS (Secure Socket layer/Transport Layer Security)

20.1.1 概述

SSL/TLS 是 Secure Socket Layer/Transport Layer Security 的缩写。SSL 协议最初由 Netscape 公司设计开发, 1994 年 11 月推出 SSL v2.0 版本。1996 年 3 月在对 SSL v2.0 进行重大改进的基础上, 推出 SSL v3.0 版本, 它解决了 SSL v2.0 中存在的许多问题, 改进了它的许多局限性并且支持更多的加密算法。IETF 于 1999 年 1 月推出 TLS v1.0 协议。TLS v1.0 是一个 Internet 标准, 完全建立在 SSL v3.0 协议的基础上, 又称 SSL v3.1 协议。TLS v1.0 解决了 SSL v3.0 存在的一些问题, 并在某些方面做了改进。其实, TLS v1.0 和 SSL v3.0 差异不是很大, 但这些差异也足以令二者不具备互操作性。然而 TLS v1.0 协议提供了支持运行 SSL v3.0 协议的机制。

SSL/TLS 协议用于在两台机器之间提供端到端的数据保密性、数据完整性服务以及通信双方的身份认证服务。SSL 协议的优势在于它是与应用层协议独立无关的, 高层的应用协议 (如 HTTP、FTP、TELNET) 能透明地建立于 SSL 协议之上。

SSL 协议位于 TCP/IP 协议模型的网络层和应用层之间, 使用 TCP 来提供一种可靠的端到端的安全服务, 它使客户/服务器应用之间的通信不被攻击窃听, 并且始终对服务器进行认证, 还可以选择对客户进行认证。SSL 协议在应用层通信之前就已经完成加密算法、通信密钥的协商以及服务器认证工作, 在此之后, 应用层协议所传送的数据都被加密。

SSL 实际上是由共同工作的两层协议组成的, 如图 20-1 所示。从体系结构图可以看出 SSL 安全协议实际是握手协议 (Handshake Protocol)、改变密码约定协议 (Change Cipher Spec Protocol)、警告协议 (Alert Protocol)、应用数据协议 (Application Data Protocol) 和记录协议 (Record Protocol) 组成的一个协议簇。

握手协议	警告协议	改变密码约定协议	应用数据协议
记录协议			
TCP			
IP			

图 20-1 SSL 协议体系结构

20.1.2 记录协议

SSL 记录协议为 SSL 连接提供了两种服务:

- ① 机密性服务: 握手协议定义了共享的、可用于对 SSL 有效载荷进行常规加密的密钥。
- ② 消息完整性服务: 握手协议还定义了共享的、可用来形成报文的鉴别码 (MAC) 的密钥。

在 SSL 协议中,所有的传输数据都被封装在记录中。记录是由记录头和长度不为 0 的记录数据组成的。所有的 SSL 通信都使用 SSL 记录协议,记录协议封装上层的握手协议、警告协议、改变密码约定协议和应用数据协议。SSL 记录协议规定了记录头和记录数据的具体格式。SSL 记录协议定义要传输数据的格式,它位于可靠的传输协议之上(如 TCP),用于各种更高层协议的封装,记录协议主要完成分组和组合,压缩和解压缩,以及消息认证和加密等功能。

SSL 记录协议的数据处理过程描述如下:

- ① 记录层从上层接收应用数据。
- ② 记录层对数据进行分段,将其分割成可管理的明文块。每个记录的长度为 16KB 或者更小。记录层中不关心客户消息的边界,属于同一内容类型的消息可以放在同一记录层报文中传输。同样,一个较大的消息也可以被分段,通过多个记录层报文来传送。
- ③ 使用当前会话状态中定义的压缩算法对明文块记录进行压缩。这一步是可选的,这里的数据块压缩是无损压缩,并且增加的长度不能超过 1024 字节。
- ④ 对第 3 步的结果,使用当前会话状态中定义的消息验证码 MAC 算法和 MAC 密钥,计算 MAC 值,并添加到第 3 步的结果之后。消息验证码主要是为了检验消息的完整性。
- ⑤ 对第 4 步产生的结果利用 IDEA、DES、3DES 或其他加密算法进行加密。加密对内容长度的增长也不可以超过 1024 字节。
- ⑥ 对第 5 步产生的结果添加记录头。记录头的作用是为接收方提供对记录进行解释所必需的信息。记录头部分包含 3 部分信息:内容类型(8 位)、SSL 协议版本号和长度域(16 位)。内容类型字段标识消息类型,SSL 支持的 4 种类型:application_data、alert、handshake 和 change_cipher_spec。长度字段可以让接收方知道它要从线路上读取多少字节才能对消息进行处理。版本号只是一项确保每一方使用所磋商版本的冗余性检查。

20.1.3 握手协议

SSL 中最复杂、最重要的部分是握手协议。这个协议用于建立会话,协商加密方法、鉴别方法、压缩方法和初始化操作,使服务器和客户能够相互鉴别对方的身份、协商加密和 MAC 算法以及用来保护在 SSL 记录中发送数据的加密密钥。在传输任何应用数据之前,都要使用握手协议。

SSL 握手协议流程如图 20-2 所示,具体的数据处理过程描述如下:

- ① 客户端将它所支持的算法列表连同同一个密钥产生过程用作输入的随机数发送给服务器。
- ② 服务器根据列表内容选择一种加密算法,并将其连同一份包含服务器公用密钥的证书发送给客户端。该证书还包含了用于认证目的的服务器标识,服务器同时还提供了一个作为密钥产生过程部分输入的随机数。
- ③ 客户端对服务器的证书进行验证,并抽取服务器的公钥。然后,再产生一个称作 pre_master_secret 的随机密码串,并使用服务器的公钥对其进行加密。最后,客户端将加密后的信息发送给服务器。
- ④ 客户端与服务器端根据 pre_master_secret 以及客户端与服务器端交换的随机数值,独立计算出加密和 MAC 密钥。
- ⑤ 客户端将所有握手消息的 MAC 值发送给服务器。

⑥ 服务器将所有握手消息的 MAC 值发送给客户端。

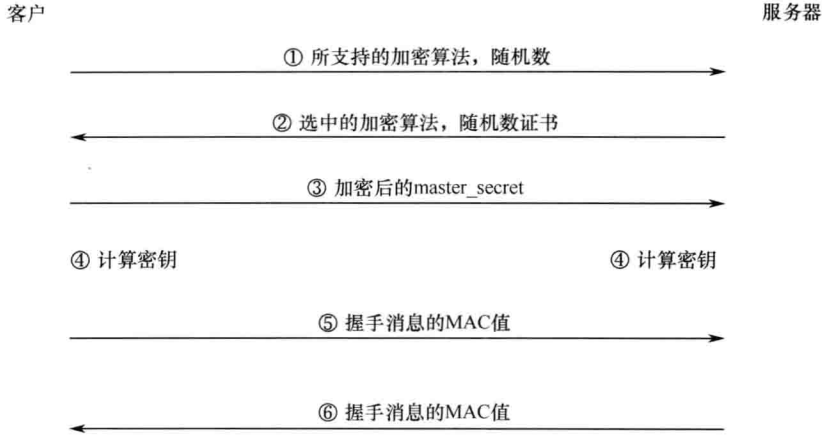


图 20-2 SSL 握手协议流程

第①步和第②步实现的目标是就一组算法达成一致。客户端告诉服务器它所支持的算法，而服务器选择其中的一种算法。当客户端收到服务器在第②步所发的消息时，它也会知道这种算法，所以双方在以后的通信中知道要使用的协议版本和算法。

第②步和第③步的目标是确立一组加密算法。第②步服务器向客户端提供其证书，这样就可以允许客户端给服务器传送密码。经过第③步后，客户端与服务器端都知道了 pre_master_secret。客户端知道 pre_master_secret 是因为这是它产生的，而服务器则是通过解密而得到 pre_master_secret 的。第③步是握手过程的关键步骤，所要保护的数据都依赖于 pre_master_secret 的安全。客户端使用服务器的公钥（从服务器证书中抽取）来加密共享密钥，而服务器使用其私钥对共享密钥进行解密。握手的剩余步骤主要用于确保这种交换过程的安全进行。

第④步中，客户端与服务器分别使用相同的密钥导出函数（Key Derivation Function，KDF）来产生 master_secret，最后再次通过 KDF 使用 master_secret 来产生加密密钥。

第⑤步与第⑥步用以防止握手本身遭受篡改。客户端提供多种算法的情况相当常见，某些强度弱而某些强度强。攻击者可以删除客户端在第①步所提供的所有高强度算法，迫使服务器选择一种弱强度的算法。第⑤步与第⑥步的 MAC 交换就能阻止这种攻击，因为客户端的 MAC 是根据原始消息计算得出的，而服务器的 MAC 是根据攻击者修改过的消息得出的，这样经过检查就会发现不匹配。由于客户端与服务器所提供的随机数为密钥产生过程的输入，所以握手不会受到重放攻击的影响。这些消息是首个在新的加密算法与密钥下加密的消息。

因此，在此过程结束时，客户端与服务器已就使用的加密算法达成一致，并拥有了一组与那些算法一起使用的密钥。更重要的是，它们可以确信攻击者没有干扰握手过程，所以磋商过程反映了双方的真实意图。

20.1.4 警告协议

SSL 警告协议是用来为对方传递 SSL 的相关警告。如果在通信过程中某一方发现任何异常，就需要给对方发送一条警示消息通告。

警示消息有两种：一种是 Fatal 错误，如传递数据过程中，发现错误的 MAC，双方就需要立即中断会话，同时消除自己的缓冲区中相应的会话记录；第二种是 Warning 消息，这种情况下通信双方通常都只是记录日志，而对通信过程不造成任何影响。SSL 握手协议可以使得服务器和客户能够相互鉴别对方，协商具体的加密算法和 MAC 算法以及保密密钥，用来保护在 SSL 记录中发送的数据。

20.1.5 改变密码约定协议

SSL 改变密码约定协议是使用 SSL 记录协议服务的 SSL 高层协议的 3 个特定协议之一，也是其中最简单的一个。协议由单个消息组成，该消息只包含一个值为 1 的单个字节。该消息的唯一作用就是将未决状态复制为当前状态，更新用于当前连接的密码组。为了保障 SSL 传输过程的安全性，双方应该每隔一段时间改变加密规则。

20.1.6 应用数据协议

SSL 应用数据协议包括常用的应用层协议，如超文本传输协议（Hyper-Text Transfer Protocol, HTTP）、文件传输协议（File Transfer Protocol, FTP）等。

20.2 IPSec

1. 概述

IPSec 协议是 IETF 于 1998 年 11 月公布的 IP 安全标准，它是在 IP 层上对数据包进行高强度的安全处理，提供包括访问控制、无连接的完整性、数据源认证、抗重播保护、保密性和有限传输流保密性在内的服务，这些服务提供对 IP 及其上层协议的保护。IPSec 提供了一种标准的、健壮的、包容广泛的、易于扩展的、完整的基础网络安全方案，目前被广泛应用于实现端到端的安全、虚拟专用网和安全隧道，是一种可为任何形式的 Internet 通信提供安全保障的协议。

IPSec 协议是一个协议簇，它包含 ESP 协议、AH 协议和 IKE 协议等，图 20-3 显示了 IPSec 的体系结构、组件及各组件间的相互关系。

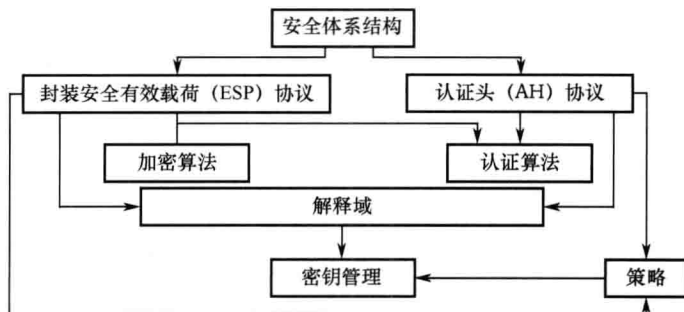


图 20-3 IPSec 协议体系结构

(1) ESP (Encapsulating Security Payload, 封装安全载荷)

ESP 机制通过将整个 IP 分组或将上层协议部分（即传输层协议数据，如 TCP、UDP 或 ICMP 协议数据）封装到一个 ESP 载荷之中，然后对此载荷进行相应的安全处理，如加

密处理、鉴别处理等，以实现通信的机密性和安全性保护。

(2) AH (Authentication Header, 验证报头)

AH 机制主要用于为通信提供完整验证性服务，还能为通信提供加密和抗重放攻击服务。

(3) 加密算法

描述各种加密算法如何应用于 ESP 中，默认的算法为 DES-CBC 算法。

(4) 验证算法

描述各种身份验证算法如何应用于 AH 协议和 ESP 协议中的身份验证选项，默认的有 HMAC-MD5 和 HMAC-SHA1 算法。

(5) 密钥管理

密钥管理的一组方案 IKE (Internet 密钥交换协议) 是默认的密钥交换协议。

(6) 解释域

彼此相关各部分的标准符及运作参数，它实际是一个存放所有 IPSec 安全参数的数据库，这些参数可被与 IPSec 服务相应的系统参考并调用。

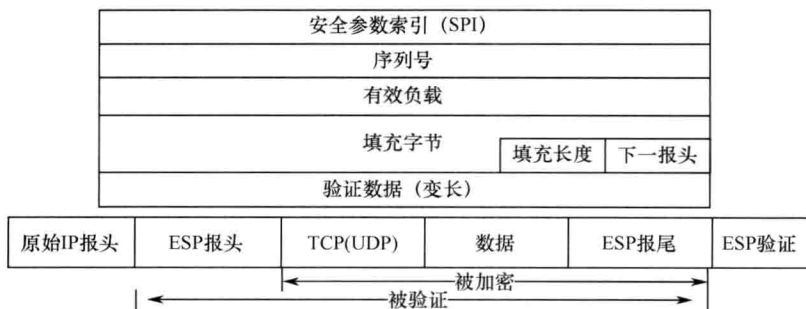
(7) 策略

它决定两个实体之间是否能够通信和如何通信，策略的核心由三部分组成：SA (Security Association)、SAD (Security Association Database)、SPD (Security Policy Database)。安全策略数据库 (Security Policy Database, SPD) 对 IPSec 策略加以维护。在 SPD 数据库中，每个条目都定义了所要保护的通信类别、保护方法以及谁共享这种保护。进入或离开 IP 堆栈的数据包都必须检索 SPD 数据库，调查可能的安全应用。每一个 SPD 条目都要定义一个对数据包的处理动作，这个动作是丢弃、透传或应用 IPSec 处理中的一种。其中，如果 SPD 项定义的动作作为应用 IPSec 处理，则会指向一个或一套安全联盟 (SA)，表示对数据包应用安全保护。SA 表示策略实施的具体细节，包括源地址、目的地址、应用协议、SPI (安全策略索引)、所用算法、密钥、长度；SAD 为进入和外出包处理维持一个活动的 SA 列表。

2. AH 协议和 ESP 协议

IPSec 协议主要使用验证报头协议 (AH) 和封装安全载荷协议 (ESP) 来提供数据的安全保证，这两个协议可以独立使用也可以组合使用以提供 IPv4 和 IPv6 环境下所需要的安全服务。

ESP (Encapsulation Security Payload) 是一个安全协议头，属于 IPSec 的一种协议，通过对原始有效负载进行加密和认证并将分组封装在一个 ESP 报头和 ESP 报尾之间，提供了对 IP 数据包的机密性、数据的完整性以及可选的数据源身份认证和抗重放攻击的服务。应用 ESP 时，要在 IP 报头之后、上层协议 (要保护的数据) 之前插入一个 ESP 头，同时在报文最后加入一个 ESP 尾将整个 IP 数据包封装起来。图 20-4 描述了传输模式下 ESP 封装 IP 报文的格式。



AH 协议是一种 IPSec 协议，用于为 IP 通信提供数据完整性、数据原始身份验证和一些可选的、有限的抗重播服务，它能保护通信免受篡改但不能防止窃听，适合用于传输非机密数据。AH 不提供任何的保密性服务，它的作用是为 IP 数据流提供高强度的密码认证，以保证数据包在被改变后能被察觉的完整性机制、验证数据包的来源的认证机制。AH 通过在整个 IP 数据包中使用一个消息验证码（MAC）来提供完整性和认证服务。一个消息验证码就是一个特定的单项散列函数，它接受一个任意长度的消息和一个密钥，生成一个固定长度的输出，称作消息摘要或指纹。MAC 不同于一般的杂凑函数，因为它是需要密钥的，最常用的 MAC 是 HMAC。IPSec 在发送数据包和接收数据包之后都可以根据一组数据计算出 MAC，其结果放到 AH 包头的验证数据区。如果两次的值一样则说明数据包没有被修改过。AH 协议具有传输和隧道两种模式和外出与进入处理方式。图 20-5 描述了传输模式下 AH 保护 IP 报文的格式。

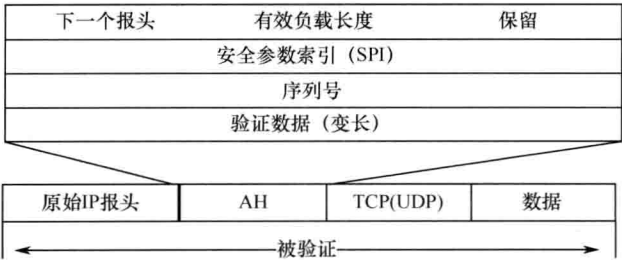


图 20-5 AH 包头及传输模式封装 IP 报文格式

3. 安全关联数据库

安全关联数据库（SAD）用于存放安全关联（SA）。安全关联（SA）是 IPSec 的基础，它决定保护什么、如何保护以及谁来保护通信数据，是两个通信实体经过协商建立起来的一种协定，规定用来保护数据安全的模式、算法及密钥、生存期、抗重放窗口、计数器等。SA 是单向的，因此两个 IPSec 对等体之间的 IPSec SA 成对出现，分别用于各自的进入和外出处理。SA 还与协议相关，每一种协议都有一个 SA。安全关联由一个三元组唯一标识：安全参数索引（SPI）、IP 目的地址及安全协议（AH 或 ESP）标识符。IPSec SA 可用 IKE 协议自动协商或进行手工设置获得，并进行有效性判断，规定的安全服务通过使用 AH 或 ESP 协议体现。

安全关联数据库（SAD）用于存放安全关联（SA），每一个 IPSec 实现都必须有一个名义上的安全关联数据库，新建立的 SA 要写入 SAD，每个 SA 在 SAD 中有一个入口。SAD 定义了主机、安全网关上实现输入、输出 IP 通信的策略。每个入口定义一个 SA 的参数，对于输出处理，SAD 入口被 SPD 中的入口指定；对于输入处理，SAD 中的每个入口由接收到的报文域中的目的 IP 地址、IPSec 协议类型和 SPI 索引来查找。

4. 安全策略数据库

IPSec 策略由 IPSec 安全策略数据库（SPD）维护，每个策略条目都在入口定义了要保护的是哪种类型的通信、如何保护以及如何共享这种保护，所有进入或外出的 IP 数据流都要检索 SPD 数据库，查找可能的安全应用。SPD 处理策略是 SA 处理过程的核心部分，它定义了提供给 IP 数据报的服务和方式，每一个 SPD 条目都定义了对输入和输出数据报文

的处理行为，是丢弃、透传或者应用 IPSec 三种中的一种。应用 IPSec 处理是数据报文在通过 IPSec 代理时进行 IPSec 处理，对于需要处理的数据报，SPD 将会确切地指出应该提供的安全服务、协议类型、应该使用的算法等信息。SPD 是 IPSec 在系统中维护的系统安全策略集，一条安全策略可以指定一个或多个 SA 策略应用于一个指定的数据报文流上，但在 SPD 中的策略项必须保存这些 SA 的顺序，在实现时必须可以为输入和输出报文定义一个 SA 的处理顺序。

SPD 对于输入、输出必须要有不同的入口，每一个 IPSec 实现必须有一个可供管理的接口，允许用户或系统管理员管理 SPD。SPD 主要包括策略入口的有序列表，每一个策略入口由一个或多个选择符标志，这些选择符定义了被这一策略入口包含的 IP 数据流、策略或者 IPSec 处理的粒度。每一个入口可能定义了丢弃、透传和应用这三种动作中的一个，如果是应用，那么该入口将会提供相应信息用来检索保护该类型数据包的 SA。如果 SA 尚未建立，那么就需要请求 IKE 与远端主机协商一个。数据包到 IPSec 策略的映射关系是由选择符决定的，选择符来源于数据包的各个字段，包括：目的地址、源地址、名字、传输层协议和源目的端口。这些选择符的值可能是一个特定取值、一个范围或者是不透明。策略的表示方式直接影响到 IPSec 的灵活性和表示能力，在实现中利用了 Internet 安全关联和密钥管理协议中提案载荷的格式来描述策略。

5. IPSec 的工作模式

IPSec 协议（包括 AH 和 ESP）可分别工作在传输模式和隧道模式，前者是用来保护 IP 报头之后的上层协议和数据，后者保护包括 IP 报头的整个 IP 报文分组。

（1）传输模式

传输模式用来保护 IP 报头之后的有效载荷，在 IP 报头和上层协议报头之间插入一个 IPSec 报头，其通信的终点必须是一个加密的终点，用于保障端到端的通信安全。

图 20-6 说明了传输模式下 IPSec 保护 IP 分组的格式。在这种模式下，IP 报头与原始 IP 分组中的 IP 报头相同，只是 IP 协议字段被改为 ESP（50）或 AH（51），并重新计算 IP 报头的校验和，IPSec 源端点不会修改 IP 报头中的目的地址。

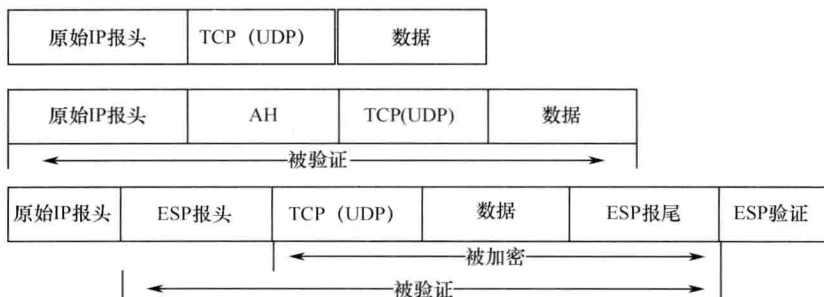


图 20-6 传输模式下 IP 报文封装格式

（2）隧道模式

隧道模式用来保护整个 IP 数据包。要保护的整个 IP 包都封装到一个新的 IP 包里，同时在外部 IP 报头和内部报头之间插入一个 IPSec 头，如图 20-7 所示。当隧道模式被网关使用时，可用来保护与其连接的子网，VPN 网络中采用的便是隧道模式。在隧道模式中，

通信的终点是由受保护的内部 IP 报头指定的地址，而 IPSec 的终点则是那些由外部 IP 报头指定的地址。IPSec 处理结束后，VPN 网关会剥离外部 IP 报头后，再将原始 IP 包转发到它最终的目的地。



图 20-7 隧道模式下 IP 报文封装格式

在隧道模式中，数据包的内部 IP 头是由主机创建的，外部 IP 头是由提供安全服务那个 IPSec 网关（既可以是主机，也可以是路由器）添加的。IPSec 支持嵌套隧道，所谓嵌套隧道就是对一个已经按隧道模式封装了的数据包再进行一次 IPSec 隧道处理，从而支持多级网络安全保护。例如，一家公司有一个安全网关，为防止竞争者和黑客的侵犯，在该公司网络内部另有一个安全网关，防止某些内部员工进入敏感子网。此时，若对一个保护网络内部的保护子网进行访问就要用到嵌套式隧道。图 20-8 描绘了 AH 和 ESP 联合嵌套时的使用方式。

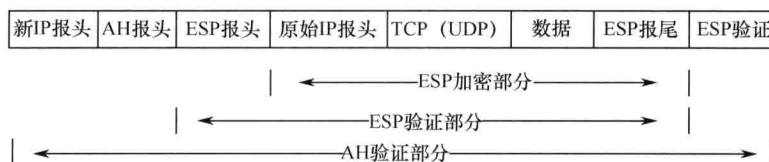


图 20-8 AH 和 ESP 联合嵌套封装 IP 报文

6. IKE 协议（Internet 自动密钥交换协议）

SA 定义了通信中应用的安全服务的细节，所以在进行 IPSec 通信之前必须在两台通信实体之间协商建立 SA。SA 分为两种：IKE SA 和 IPSec SA。对等体之间的 IKE SA 用于对控制数据流的保护（如 IPSec 的协商报文），协商对 IKE 数据流进行加密以及对对等体进行认证的算法，对等体之间只有一个 IKE SA。IPSec SA 用于协商对等体之间的 IP 数据流进行加密的算法，对哪些数据流进行加密由策略定义决定。IPSec SA 是单向的，因此它总是成对出现的，对等体之间可以有多个 IPSec SA，用于保护不同的 IP 主机组或者 IP 数据流。

SA 可以手工建立，但很明显这种情况适合于规模小、机器分配相对固定的环境。因此 IETF 定义的主要功能是自动确定和维护 IKE SA 和 IPSec SA 的 IKE 协议。IKE 分两个阶段运行，以分别确定 ISAKMP SA 和 IPSec SA。

阶段 1：IKE 对等体互相验证对方身份并确定回话密钥。这个阶段通过使用 DH 交换、cookie 和 ID 载荷交换创建一个 ISAKMP SA（也叫 IKE SA）。之后，发起方和响应方之间的所有 IKE 通信都将通过 IKE SA 进行加密和完整性保护。IKE 的第一阶段主要是在安全对等体之间建立一条安全信道，以便第二阶段协商能够安全进行。阶段 1 的协商模式包括主模式和野蛮模式。

阶段 2：主要进行协商 SA 和协商用于保护用户数据的密钥协商（如 IPSec SA），阶段 2 的协商消息报文是在阶段 1 IKE SA 的保护下进行的。阶段 2 的模式为快速模式。

20.3 Kerberos

1. 概述

Kerberos 身份认证服务是目前较为著名、也相对较为成熟的一种身份认证机制。Kerberos 协议是一种应用于开放式网络环境，基于可信任第三方和 TCP/IP 的网络安全认证协议，其认证模型基于 Needham-Schroeder，以加密为基础，对用户及网络连接进行认证，提供增强网络安全的服务。

该协议是美国麻省理工学院（MIT）“雅典娜项目”（Project Athena）的一部分，使用 DES 来进行加密和认证。至今，Kerberos 协议已经有了 5 个版本，Kerberos v4 是被公诸于众的第一个版本，它是分布式计算机环境框架的组成部分，已在一些 UNIX 系统中得到应用。

Kerberos 的设计是针对以下要求实现的：

- ① 安全性：网络中的窃听者不能获得必要的信息来假冒网络的用户。
- ② 可靠性：对基于 Kerberos 协议的所有服务来说，Kerberos 系统的瘫痪则意味着它所支持的所有服务的瘫痪。
- ③ 透明性：用户除了被要求输入密码外，不会觉察出认证的进行过程。
- ④ 可扩充性：系统应能够支持更多数据的用户和服务。

Kerberos 的设计目的分为 3 个领域：认证、授权、记账，可用于构建安全网络，它提供了可用于安全网络环境的认证机制和加密工具。Kerberos 协议已成为业界的标准网络身份验证协议，得到了越来越广泛的应用。微软（Microsoft）公司操作系统 Windows 2000 中提供了对 Kerberos v5 的支持，Linux Redhat 环境下也可以使用 Kerberos 提供的 Ktelnetd、Krlgind、Krsd 来替代传统的 telnetd、rlogind、rsd 服务，Java 安全解决方案中的 Java 认证和权限服务（JAAS, Java Authentication and Authorization Service）也提供了对 Kerberos 的支持。

Kerberos 的目标是将认证从不安全的工作站转移到集中的认证服务器（Authentication Server），服务器在物理上是安全的，并且其可靠性也是可控制的。必须保证用户的密码不能以明文形式在网络中传输，每个用户和服务都是一个密码，只有认证服务器拥有所有用户和服务的密码。

2. 基本原理

Kerberos 协议通过提供认证中心服务，并应用对称密码体制 DES，在客户机和服务器之间构造起了一个安全桥梁。针对用户向服务器提交的每个服务请求及其权限，要求用户必须预先经过认证中心服务器的认证合格后，才能被指定服务器所执行，这样在很大程度上消除了许多潜在的安全问题。

Kerberos 协议是以可信赖第三方为基础的认证协议，有域内认证和域间认证两种模式。域间认证模式的基本原理与域内认证类似。如果客户端要访问不在同一个域内的服务器，就必须从另一个域的 TGS 获得该服务器的票据。在两个 Kerberos 域之间建立信任关系，域之间通过这两个域间的密钥来加密之间传送的数据。以后的过程则与域内认证过程相同。

域内认证模型认证过程如图 20-9 所示。

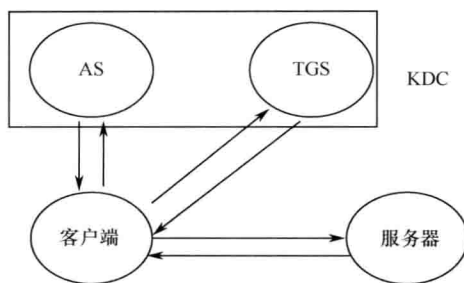


图 20-9 域内认证流程图

图中所用到的符号说明如下。

Client: 客户端 (C)。

KDC: 密钥分配中心 (Key Distribution Center)，由 AS 和 TGS 组成。

AS: 认证服务器 (Authentication Server)。

TGS: 票据授权服务器 (Ticket Granting Server)。

Server: 应用服务器 (S)。

K_c : 客户端 C 的密钥。

K_{tgs} : TGS 的密钥。

K_s : 应用服务器 S 的密钥。

$K_{c, tgs}$: 客户端 C 和 TGS 共享的会话密钥。

TGT: 票据授权票，用于访问 TGS 的票据。

在 AS 服务器中，保存有 K_c 、 K_{tgs} 和 K_s 密钥，AS 服务器分别与客户端、TGS 服务器和应用服务器共享这些密钥。

AS 用于在登录时验证用户身份，TGS 用于发放身份证明票据。

Kerberos 中使用两种凭证：票据 (Ticket) 和认证单 (Authenticator)。将票据发送给服务器时，票据用来安全地传送客户的身份。票据中有一些信息，服务器可用这些信息保证使用票据的客户就是拥有票据的客户。认证单是与票据一起呈交的其他凭证。

Kerberos 协议在很大程度上依赖于共享密钥的认证技术。它的基本概念很简单：如果一个密码只有两方知道，那么双方都能通过证实另一方是否知道该密码来验证其身份。但这种方式存在一个问题，就是通信双方如何知道这一共享的密码，如果通过网络传输密码，则很容易被网络监听器所截获和解密。对于这个问题，Kerberos 协议利用密钥加密的方法来解决。通信双方不再共享口令，而是共享一个密钥，通信双方用这个密钥的有关信息互相证实身份，而密码本身不在网络中进行传输。

共享密钥解决了密码传输的问题，但另一个问题是通信中的双方如何获取密钥。在 Kerberos 中，密钥的分配是通过 KDC 的服务来统一管理的。KDC 是一个运行在域服务器上的服务，它维护着在其管辖的域范围内所有安全主体的账号信息数据。

客户端 (Client) 要访问应用服务器 (Server)，需要进行 6 次协议交换。具体认证过程如下：

第一阶段 (认证服务交换) 客户端从 AS 处获取 TGT。

(1) Client→AS

客户端向 AS 发出访问 TGS 的请求, 请求以报文形式发送。请求报文包括客户的名称, TGS 的名称, 客户端的 IP 地址, 以及时戳。时戳用于向 AS 表示这一请求是新的。请求报文以明文方式发送。

(2) AS→Client

AS 收到客户请求报文后, 在其数据库中查找客户的加密密钥 K_c , 并产生随机会话密钥 $K_{c, tgs}$ 和 TGS 的票据 TGT 作为应答报文。会话密钥 $K_{c, tgs}$ 用于客户端与 TGS 进行加密通信, 用 K_c 加密。TGT 的内容包括: TGS 的名字, 客户名字, 客户的 IP 地址, 时戳, 有效生存期限, 以及会话密钥 $K_{c, tgs}$ 。这些数据使用 TGS 的密钥 K_{tgs} 进行加密, 以保证只有 TGS 才能解密。

AS 向客户端发出应答。应答内容用客户的密钥 K_c 加密, 使得只有客户端 Client 才能解密该报文的内容。如果客户端解不开该应答报文, 则表明该客户的身份是假冒的, 身份认证宣告失败; 如果客户端能解开该报文, 则证明该客户身份是正确的。

客户端收到 AS 返回的应答报文后, 在本地输入口令生成 K_c , 对报文进行解密, 就得到 TGS 的票据 TGT, 客户在下一步就可以把 TGT 发送给 TGS 来证明自己具有访问 TGS 的合法身份。客户端同时从 AS 处得到了与 TGS 的会话密钥 $K_{c, tgs}$, 用它来与 TGS 进行加密通信。

第二阶段(授权服务交换)客户从 TGS 处获取访问应用服务器的票据 $T_{c, s}$ 。

(3) Client→TGS

客户端向 TGS 发送访问应用服务器 S 的请求报文, 报文内容包括要访问的应用服务器 S 的名字, TGS 的票据 TGT 以及认证符 $A_{c, tgs}$ 。

TGT 的内容是用 TGS 的密钥 K_{tgs} 加密的, 只有 TGS 才能解开。认证符的内容包括客户的名字、客户的 IP 地址以及时戳, 认证符的内容用客户和 TGS 的会话密钥进行加密, 以保证只有 TGS 才能解开。票据 TGT 可以重复使用且有效期较长, 而认证符只能使用一次且有效期很短。

TGS 收到客户端发来的请求报文后, 用自己的密钥 K_{tgs} 对票据 TGT 进行解密处理, 得知客户已经从 AS 处得到与自己的会话密钥 $K_{c, tgs}$, 此处票据 TGT 的含义为“使用 $K_{c, tgs}$ 的客户是 C”。TGS 用 $K_{c, tgs}$ 解密认证符, 并将认证符中的数据与 TGT 中的数据进行比较, 从而可以相信 TGT 的发送者 C 就是 TGT 的实际持有者。

此处的票据 TGT 并不能证明任何人的身份, 只是用来安全地分配密钥, 而认证符则用来证明客户身份。因为认证符只能被使用一次且其有效期很短, 所以可以防止对票据和认证符的盗窃使用。

(4) TGS→Client

TGS 检验认为客户身份合法以后, 产生随机会话密钥 $K_{c, s}$, 该密钥用于客户端 C 和应用服务器 S 进行加密通信, 同时产生用于访问应用服务器 S 的票据 $T_{c, s}$ 。 $T_{c, s}$ 的内容包括: 应用服务器的名字, 客户的名字, 客户的 IP 地址, 时戳, 有效生存期和会话密钥 $K_{c, s}$ 。 $T_{c, s}$ 的内容用应用服务器 S 的密钥 K_s 加密, 以保证只有 S 才能解开。会话密钥 $K_{c, s}$ 和票据 $T_{c, s}$ 组成 TGS 的应答报文, 该应答报文用客户 C 和 TGS 的会话密钥 $K_{c, tgs}$ 加密, 以保证只有客户端 C 才能解开。TGS 将该应答报文发送给客户端 C。

客户端 C 收到 TGS 的应答报文后, 用会话密钥 $K_{c, tgs}$ 对报文进行解密, 可以得到访问应用服务器 S 的票据 $T_{c, s}$, 以及与 S 进行加密通信的会话密钥 $K_{c, s}$ 。只有合法的用户 C 才能解开给报文的内容。

第三阶段 (客户端/应用服务器交换) 客户端与服务器相互验证身份。

(5) Client→S

客户端 C 向应用服务器 S 发送请求报文, 报文的内容包括应用服务器的名字, 用于访问应用服务器 S 的票据 $T_{c, s}$, 以及认证符。

$T_{c, s}$ 的内容是用应用服务器 S 的密钥 K_s 加密的, 只有 S 才能解开。认证符的内容包括客户的名字, 客户的 IP 地址, 时戳。认证符的内容用客户和服务器的会话密钥加密, 以保证只有应用服务器 S 才能解开。票据 $T_{c, s}$ 可以重复使用且有效期较长, 而认证符只能使用一次且有效期很短。

应用服务器 S 收到客户端发来的请求报文后, 用自己的密钥 K_s 对票据 $T_{c, s}$ 进行解密处理, 得知客户 C 已经从 TGS 处得到与自己的会话密钥 $K_{c, s}$ 。此处票据 $T_{c, s}$ 的含义为“使用 $K_{c, s}$ 的客户端是 C”。S 用 $K_{c, s}$ 解密认证符, 并将认证符中的数据与 $T_{c, s}$ 中的数据进行比较, 从而可以相信 $T_{c, s}$ 的发送者 C 就是 $T_{c, s}$ 的实际持有者, 客户端 C 的身份得到验证。

(6) S→Client

应用服务器 S 检验认为客户端 C 身份合法后, 对从认证符中得到的时戳加 1, 然后用与客户端共享的会话密钥 $K_{c, s}$ 加密后作为应答报文发给客户。该应答报文只有客户 C 才能解开。

客户 C 收到应用服务器 S 发来的应答报文后, 用会话密钥 $K_{c, s}$ 进行解密后, 对应答报文中增加的时戳进行验证, 通过比较时间戳的有效性实现对 S 的认证。验证正确则相信增加时戳的确实是应用服务器 S, 从而应用服务器 S 的身份得到验证。

整个协议交换过程结束后, 客户和应用服务器之间就拥有了共享的会话密钥, 双方以后就可以用该会话密钥进行加密通信了。

20.4 TSP

1. 简介

在书面合同中, 与手写签名一样, 签署日期也是一项关键性内容, 用于防止文件被伪造和篡改。在电子交易时代, 如何确定电子文件的签署日期并防止被篡改成为一个难题。时间戳技术 (TSP, Time-Stamp Protocol) 能有效解决电子文件的签署日期问题, 通过数字签名技术将电子数据和特定时间绑定在一起, 既能准确确定电子文件的签署日期, 又能有效防止该签署日期和文件内容被伪造或修改, 实质上是数字签名技术的一种特殊应用。

通过数字签名技术将电子数据和特定时间绑定后的结果称作时间戳 (time-stamp), 它主要由 3 部分内容组成: 电子数据的摘要值、特定时间、数字签名。

书面合同的时间是签署人自己写上的, 但时间戳则不然, 其中的特定时间和数字签名

均由特定机构 TSA (Time Stamping Authority) 签署。TSA 可理解为可信第三方 TTP (Trusted Third Party) 服务机构, 具有权威性, 专门为用户数据签发时间戳。

时间戳技术总体结构如图 20-10 所示。

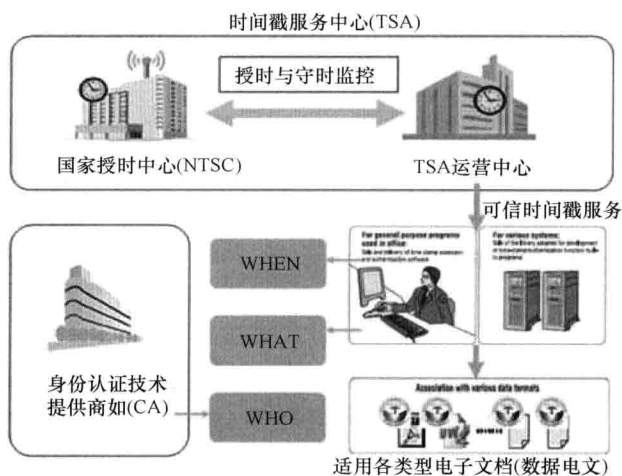


图 20-10 时间戳总体结构

时间戳产生的过程主要包括以下几个步骤:

① 用户(时间戳需求方)将电子数据(文件)使用摘要算法计算出摘要值, 然后组成时间戳请求包 TimeStampReq。

② 用户将请求包 TimeStampReq 发送给 TSA。

③ TSA 接收到请求包 TimeStampReq。

④ TSA 需要验证 TimeStampReq 的时效性, 通过判断 nonce 是否重复来防止重放攻击。TSA 使用私钥对请求包中的摘要进行数字签名后, 组成时间戳响应包 TimeStampResp。TSA 可以拥有多个私钥, 针对不同策略、不同算法等, 可使用不同的私钥。TSA 数字证书的 extendedKeyUsage 扩展项必须设置为关键扩展项, 且必须包含 id-kp-timeStamping 扩展密钥用途。

⑤ TSA 将响应包 TimeStampResp 发送给用户。

⑥ 用户接收到响应包 TimeStampResp。

⑦ 用户首先判断 TimeStampResp 中的状态信息, 如果为错误状态, 则表示本次时间戳申请失败; 如果为正确状态, 则验证响应包中各种字段信息和 TSA 签名是否正确; 如果字段信息或签名不正确, 则拒绝该响应包。用户需要通过 OCSP 或 CRL 验证 TSA 证书是否作废或有效。

2. 请求包格式

时间戳请求包格式用 ASN.1 描述如下:

```
TimeStampReq ::= SEQUENCE {
    version                INTEGER { v1(1) },
    messageImprint
```

```

--a hash algorithm OID and the hash value of the data to be time-stamped
reqPolicy      TSAPolicyId      OPTIONAL,
nonce          INTEGER          OPTIONAL,
certReq        BOOLEAN          DEFAULT FALSE,
extensions     [0] IMPLICIT Extensions OPTIONAL }

```

(1) version

version 字段用于区分请求包格式的版本，缺省值为 v1 (0)。

(2) messageImprint

messageImprint 字段包含待签署时间戳的数据的摘要值，其格式用 ASN.1 描述如下：

```

MessageImprint ::= SEQUENCE {
    hashAlgorithm      AlgorithmIdentifier,
    hashedMessage      OCTET STRING }

```

如果 TSA 不能识别请求包中的摘要算法 messageImprint→hashAlgorithm，或者 TSA 认为请求包中的摘要算法强度不够，TSA 将拒绝提供时间戳服务，并在响应包 PKIStatusInfo 中返回状态 badAlg（表示算法错误）。

(3) reqPolicy

若 reqPolicy 字段存在，则表示响应包中 TimeStampToken 字段应遵循该策略。TSAPolicyId 用 ASN.1 描述如下：

```
TSAPolicyId ::= OBJECT IDENTIFIER
```

(4) nonce

当客户端（请求包发起者）本地没有时钟时，通过 nonce 字段来验证响应包的时效性，表示与请求包对应的响应包中必须包含相同的 nonce，否则客户端拒绝该响应包。

nonce 是个大整数（如 64 位长整数），由客户端随机产生，尽量避免重复。

(5) certReq

如果 certReq 字段存在且设置为 true，则表示 TSA 公钥证书必须包含在响应包的 SignedData 中的 certificates 字段中。

如果 certReq 字段不存在或设置为 false，则表示响应包中 SignedData 中 certificates 字段不应该存在。

(6) extensions

extensions 字段用于请求包中的信息扩展，可包含多个扩展信息。extensions 格式同 X.509 数字证书扩展项格式。

如果 TSA 无法识别请求包中的扩展项，不管其是关键扩展项还是非关键扩展项，TSA 都不应该签发时间戳，应该在响应包 PKIStatusInfo 中返回状态 unacceptableExtension（表示扩展项未知）。

3. 响应包格式

时间戳响应包格式用 ASN.1 描述如下：

```

TimeStampResp ::= SEQUENCE {
    status          PKIStatusInfo,
    timeStampToken  TimeStampToken OPTIONAL }

```

(1) status

status 字段表示返回状态，其格式用 ASN.1 描述如下：

```
PKIStatusInfo ::= SEQUENCE {
    status          PKIStatus,
    statusString    PKIFreeText    OPTIONAL,
    failInfo        PKIFailureInfo OPTIONAL }
```

① status: 当 status 为 0 或 1 时，timeStampToken 必须存在。当 status 为其他值时，timeStampToken 不应该存在。常用 status 值用 ASN.1 描述如下：

```
PKIStatus ::= INTEGER {
    granted          (0),    --表示成功
    grantedWithMods (1),    --表示成功但需修改
    rejection        (2),    --表示拒绝
    waiting          (3),    --表示等待
    revocationWarning (4),   --表示即将作废
    revocationNotification (5) --表示已经作废
}
```

② failInfo: failInfo 表示 TSA 拒绝签发时间戳的原因代码，此时 TimeStampToken 字段不存在。常用 failInfo 值用 ASN.1 描述如下：

```
PKIFailureInfo ::= BIT STRING {
    badAlg          (0),    --表示算法不支持或不识别
    badRequest      (2),    --表示请求包错误
    badDataFormat   (5),    --表示提交的数据格式错误
    timeNotAvailable (14),   --表示 TSA 时间源不可用
    unacceptedPolicy (15),   --表示请求包中策略不支持
    unacceptedExtension (16), --表示请求包中扩展项不支持
    addInfoNotAvailable (17), --表示扩展项中额外信息不支持
    systemFailure    (25)   --表示系统故障
}
```

③ statusString: statusString 用于表示 TSA 拒绝签发时间戳的原因描述，如“messageImprint 字段格式错误”。

(2) timeStampToken

timeStampToken 表示时间戳令牌，用 ASN.1 描述如下：

```
TimeStampToken ::= ContentInfo
ContentInfo ::= SEQUENCE {
    contentType    ContentType,
    content        [0] EXPLICIT ANY DEFINED BY contentType OPTIONAL }
ContentType ::= OBJECT IDENTIFIER
```

其中，contentType 为 signedData。content 定义为 SignedData 类型。其中，SignedData→contentInfo→contentType 设置为 id-ct-TSTInfo，SignedData→contentInfo→content 定义为

TSTInfo 类型, TSA 公钥整数必须包含在 SignedData→certificates 中。用 ASN.1 描述如下:

```
SignedData ::= SEQUENCE {
    version Version,
    digestAlgorithms DigestAlgorithmIdentifiers,
    contentInfo ContentInfo,
    certificates [0] IMPLICIT ExtendedCertificatesAndCertificates OPTIONAL,
    crls [1] IMPLICIT CertificateRevocationLists OPTIONAL,
    signerInfos SignerInfos }
id-ct-TSTInfo OBJECT IDENTIFIER ::= { iso (1) member-body (2) us (840) rsadsi (113549)
pkcs(1) pkcs-9(9) smime(16) ct(1) 4}
TSTInfo ::= SEQUENCE {
    version                INTEGER { v1 (1) },
    policy                 TSAPolicyId,
    messageImprint         MessageImprint,
    serialNumber           INTEGER,
    genTime                GeneralizedTime,
    accuracy               Accuracy OPTIONAL,
    ordering               BOOLEAN DEFAULT FALSE,
    nonce                 INTEGER OPTIONAL,
    tsa                   [0] GeneralName OPTIONAL,
    extensions             [1] IMPLICIT Extensions OPTIONAL }
```

① version 字段用于区分时间戳令牌的版本, 缺省值为 v1 (0)。

② policy 字段表示产生时间戳令牌时应遵循的策略, 必须与时间戳请求包 reqPolicy 字段相同。如无法遵循 reqPolicy 策略要求, 则在响应包 PKIStatusInfo 中返回错误状态 unacceptedPolicy (表示策略无法接受)。常用 policy 策略包括时间戳令牌产生的条件、是否保留时间戳令牌日志 (用于事后验证) 等。policy 格式同请求包中的 reqPolicy 字段。

③ messageImprint 字段包含待签署时间戳的数据的摘要值, 必须与时间戳请求包 messageImprint 字段相同。messageImprint 格式同请求包中的 messageImprint 字段。

④ serialNumber 字段表示时间戳令牌的唯一序列号, 由 TSA 统一分配, 通过 TSA 名称和时间戳令牌序列号可以唯一标识时间戳令牌。serialNumber 应该支持大整数, 长度可达 160 位。

⑤ genTime 字段表示时间戳令牌的产生时间。genTime 可采用两种格式: UTCTime 和 GeneralizedTime。UTCTime 用 2 位数字表示年份, GeneralizedTime 用 4 位数字表示年份。2049 年以前的时间可采用 UTCTime 格式, 但 2050 年及以后的时间必须采用 GeneralizedTime 格式。当采用 UTCTime 格式时, 如 2 位年份数字 YY 小于 50 时, 则年份应该解释为 20YY 年; 当 YY 大于或等于 50 时, 年份应该解释为 19YY 年。

⑥ accuracy 字段表示 TSA 的时间精度。将 genTime 字段与该精度相加, 即可获得时间戳令牌产生的最大时间; 将 genTime 字段与该精度相减, 即可获得时间戳令牌产生的最小时间。当 accuracy 字段不存在时, 可通过其他方式获得 TSA 的时间精度, 如 TSAPolicyId。

accuracy 用 ASN.1 描述如下:

```
Accuracy ::= SEQUENCE {
    seconds    INTEGER              OPTIONAL,
    millis     [0] INTEGER (1..999) OPTIONAL,
    micros     [1] INTEGER (1..999) OPTIONAL }
```

其中, millis 表示毫秒, micros 表示微秒。

⑦ ordering 字段表示时间戳令牌产生时间的排序状态。如果 ordering 设置为 true, 则表示同一个 TSA 签发的所有时间戳令牌, 其 genTime 的时间顺序就表示产生顺序。即两个时间戳令牌 A 和 B, 如果 $A \rightarrow \text{genTime}$ 大于 $B \rightarrow \text{enTime}$, 则表示 B 先于 A 产生。如果 ordering 不存在或者设置为 false, 则 genTime 的时间顺序不一定表示产生顺序; 只有两个时间戳令牌的产生时间 genTime 之差大于其精度之和时, genTime 的时间顺序才能表示产生顺序。即两个时间戳令牌 A 和 B, 如果 $A \rightarrow \text{enTime}$ 与 $B \rightarrow \text{enTime}$ 之差大于 $A \rightarrow \text{accuracy}$ 与 $B \rightarrow \text{accuracy}$ 之和, 此时才表示 B 先于 A 产生, 否则无法判断 A 和 B 的产生顺序。

⑧ nonce 字段用于客户端 (请求包发起者) 验证响应包的时效性。必须与时间戳请求包 nonce 字段相同。

⑨ tsa 字段表示 TSA 名称, 可以是 TSA 证书中的 subject 字段, 也可以是 subjectAltName 扩展项。其实, TSA 证书也包含在 SignedData \rightarrow signerInfos 中。

⑩ extensions 字段用于请求包中的信息扩展, 可包含多个扩展信息。extensions 格式同 X.509 数字证书扩展项格式。

20.5 SET

SET 是 Secure Electronic Transaction (安全电子协议) 的缩写。它是 VISA 和 MasterCard 公司于 1997 年 5 月开发的一套规范, 主要目的是保证互联网在线交易时信用卡支付的安全性, 得到了 IBM、HP、Microsoft、Netscape、Verifone、GTE、Verisign 等公司的支持, 已成为事实上的工业标准, 并且获得了 IETF 组织的认可。

SET 协议的基本设计目标是保证持卡人、商家以及收单行之间在互联网上能够安全地进行支付交易。

SET 协议建立在以下几个商业要求的基础之上:

- ① 为支付信息提供机密性, 保证与支付信息同时传输的订购信息的机密性。
- ② 保证所有传输数据的完整性。
- ③ 为持卡人提供认证, 保证一个持卡人是一个支付账户的合法用户。
- ④ 为商家提供认证, 保证商家通过一个收单行可以接受该品牌的支付卡交易。
- ⑤ 保证使用最好的安全技术和系统设计来保护所有电子商务交易的合法参与者。
- ⑥ 创建一个不依赖于传输安全机制的协议。
- ⑦ 鼓励网络和软件提供商支付互操作性。

SET 规范主要包括 SET 证书管理、SET 支付系统、SET 协议的外部接口指引 3 个方面的内容。

SET 协议中定义的参与者包括持卡人、商家、发卡行、收单行、支付网关和证书授权机构，具体描述如下。

① 持卡人 (Cardholder): 在电子商务环境中, 是指信用卡和数字证书的持有者, 通过相应软件, 可以借助支付卡和数字证书与商家完成支付交易。

② 商家 (Merchant): 是指能够为持卡人提供服务或商品的实体。SET 协议中定义的商家能够与持卡人进行安全的电子交易, 并且一个商家必须与相关的收单行达成协议, 保证可以接受支付卡付款。

③ 发卡行 (Issuer): 是指金融机构为持卡人建立一个账户并发放信用卡, 保证对经过授权的交易进行付款。此外, 发卡行还负责为持卡人颁发数字证书, 数字证书用来鉴别持卡人的身份, 证书中包括关于标识持卡人持有的信用卡信息。

④ 收单行 (Acquirer): 是指为商家建立一个账户并处理信用卡授权和支付的金融机构。

⑤ 支付网关 (Payment Gateway): 是一个由收单行或指定的第三方操作的设备, 位于商家和收单行之间, 连接 SET 和现有的银行支付网络, 用于处理信用卡授权和支付。因此, 通常商家是与发卡行的支付网关进行交互, 而不是与发卡行直接进行交互。

⑥ 认证机构 (Certificate Authority): 是负责为持卡人、商家和支付网关签发和管理数字证书, 让持卡人、商家和支付网关之间可以通过数字证书进行认证的一个机构。

SET 协议的主要功能包括:

① SET 协议位于应用层, 对网络上其他各层也有涉及。它规范了整个商务活动的流程, 从持卡人到商家、支付网关、认证中心以及信用卡结算中心之间的信息流走向和必须采用的加密、认证都制定了严格的标准, 从而最大限度地保证了商务性、服务性、协调性和集成性。

② SET 是一个非常复杂的协议, 它详尽而准确地反映了参与交易的各方之间存在的各种关系, 还定义了加密信息的格式和完成一笔支付交易过程中各方传输信息的规则。

③ SET 不只是一个技术方面的协议, 它还说明了各方所持有的数字证书的合法含义和希望得到数字证书以及响应信息的各方应有的动作, 以及与一笔交易紧密相关的责任。

④ 就付款方式的实现而言, 在 SET 协议中, 商家在收到客户的信用卡及订单后, 信用卡的信息通过支付网关自动转到传统的金融网络, 在得到发卡机构的核准后银行即可进行付款。

SET 协议规定的工作流程分以下 3 个阶段:

① 在购买请求阶段, 持卡人选购商品, 确定支付方式, 向商家发送购货单和一份经过签名、加密的信托书。信托书的信用卡号是经过加密的, 商家无法获得。

② 在支付确认阶段, 商家把信托书传送到收单银行, 收单银行可以解密信用卡号, 并通过认证验证签名。收单银行向发卡银行查问, 确认持卡人的信用卡是否属实。属实则发卡银行认可并确认该笔交易, 收单银行认可商家并确认此交易, 最后商家向客户传送货物和收据。

③ 在收款阶段, 交易成功, 商家向收单银行出示所有交易的细节索款, 收单银行按合同将货款划给商家。发卡银行向用户定期寄去信用卡消费账单。

SET 协议在一般使用环境下的工作步骤如下:

① 持卡人利用电子商务平台选择物品, 并提交订单。

② 商家接收订单，生成初始应答消息，数字签名后与商家证书持卡人联系。

③ 持卡人对应答信息进行处理，选择支付方式，确认订单，签发付款指令，将订单信息和支付信息进行双重签名，它对双重签名后的信息和用支付网关公钥加密的支付信息签名后连同自己的证书发送给商家（商家看不到持卡人的账户信息）。

④ 商家验证持卡人证书和双签名后，生成支付认可请求，并连同加密的支付信息转发给支付网关。

⑤ 支付网关解密后获得用户卡信息，向发卡行请求验证持卡人的账户信息，并生成支付认可消息，数字签名后发给商家。

⑥ 商家收到支付认可消息后，验证支付网关的数字签名，生成购买订单确认信息并发送给持卡人。

⑦ 至此交易过程结束。商家发送货物或提供服务并请求支付网关将购物款从发卡银行持卡人账户转账到收单银行商家账户，支付网关完成转账后，生成取款应答消息发送给商家。

20.6 3-D Secure

SSL 协议能解决交易两端信息传输的安全问题，但无法满足电子交易支付的个性化需求；SET 协议能有效解决电子交易支付的安全性，但因其协议过于复杂、成本过高而始终无法得到广泛应用。2001 年 VISA 推出了一种能够弥补 SSL 和 SET 不足的“VISA 验证”服务，这项服务采用全球互通付款的“3-D Secure 技术”，可有效减少信用卡在网络被盗刷的风险，对持卡人、特约商家及发卡银行都是皆大欢喜的多赢结果。

3-D Secure 是 3 Domain Secure 的缩写，是 VISA 为提高电子商务支付的有效性而提出的一种认证技术，它主要采用 SSL 加密技术和商家服务器插件 MPI（Merchant Server Plug-In）技术来实现。在在线交易中，它既能查询并鉴别持卡人的身份，又能够保护支付卡信息在网络中传递的安全性。

3-D 是指三方域模型，由三个域组成，即发卡域（Issuer Domain）、收单域（Acquirer Domain）和协作域（Interoperability Domain）。由于 3-D 是由 VISA 提出的，所以在协议定义中与 VISA 服务的各个部分（如 VISA 网络、VISA 目录等）联系紧密。

发卡域主要负责用户注册以及在交易中验证注册用户并为合法用户授权。发卡行系统主要通过一个充当中介的中间目录服务器与 3-D 特约商家联系，它必须有能力同时处理多个用户通过浏览器访问互联网进行交易的操作。而持卡人不需要任何特殊的软件，他们一旦注册，就可以通过标准浏览器进行交易活动。发卡域具体包括以下几个部分：

① 持卡人（Cardholder）：持卡人是在线交易的买方。在线交易结算时，直接或通过电子钱包提供卡号、卡的有效期以完成交易。当弹出认证 Web 页面时，持卡人提供认证信息（如密码或个人确认消息）即可。

② 持卡人浏览器（Cardholder Browser）：持卡人的浏览器充当了商家服务器插件（位于收单域）和访问控制服务器（位于发卡域）之间的消息传递通道。

③ 附加的持卡人组件（Additional Cardholder Components）：其他一些可选的持卡人端的软硬件设备，如智能卡需要的专用读卡软件和读卡器。

④ 发卡行 (Issuer): 发卡行是指为持卡人建立一个账户并发放银行卡的金融机构。

⑤ 访问控制服务器 ACS (Access Control Server): 主要有两个功能, 一是验证请求支付的某一个卡号是否在被允许参与 3-D 服务的范围内 (通过注册取得参与 3-D 交易的权利); 二是对某一笔交易的持卡人进行认证或在认证无效时提供认证的证据。

收单域负责定义一个过程, 以保证参与 3-D 交易的商家所有操作符合收单行的规定。收单行还要为合法交易提供具体的交易处理。收单域主要包括以下几个部分:

① 商家 (Merchant): 通过已安装的交易软件处理交易过程, 包括获取支付卡卡号、调用 MPI 引导进入支付认证过程、认证后向收单行提交授权请求等。

② 商家服务器插件 MPI (Merchant Server Plug-In): 创建和处理支付认证消息, 并为商家交易软件返回相应的控制消息, 验证这个控制消息里的数字签名。

③ 验证过程 (Validation Process): 确认从 ACS 返回消息的数字签名, 此过程也可以由 MPI 或其他机构完成。

④ 收单行 (Acquirer): 金融机构成员。在 3-D 服务中, 主要是与商家建立契约关系并为它们承兑支付卡, 判断商家是否有资格参与 3-D 服务。

协作域利用一般的通信协议联系发卡域和收单域, 并且共享 VISA 目录和 VisaNet 网络服务, 使得整个支付流程能够顺利进行。协作域主要包括以下几个部分:

① 目录服务器 (Directory Server): 每一次支付过程都通过该目录服务器接收商家查询某支付卡卡号的请求消息, 判断这个卡号是否在合法的交易卡号范围内。将持卡人账户认证信息提交给适当的 ACS。ACS 的响应可以直接返回给商家, 也可以由目录服务器接收认证响应消息, 并由它返回给商家。

② 商业认证机构 (Commercial Certificate Authority): 为使用 3-D 服务的实体发放特定的证书, 包括 TLS/SSL 客户端和服务器证书等。

③ 方案认证机构 (Scheme Certificate Authority): 为使用 3-D 服务的实体发放特定的证书, 包括数字签名证书、支付方案所需的根证书等。

④ 认证历史服务器 (Authentication History Server): 每一次支付过程都通过它接收向 ACS 发出的支付认证请求消息和认证结果 (不管认证是否成功), 并存储和记录这个消息。当收单行和发卡行发生争执时, 可通过该服务器的数据记录进行仲裁。

⑤ 授权系统 (Authorization System): 在支付认证通过后, 授权系统 (如 VisaNet) 开始执行它的传统功能。从收单行接收授权请求, 将这些请求提交给发卡行, 将发卡行的响应返回给收单行, 提供发卡行和收单行之间的清算服务。

根据规范, 3-D Secure 的工作流程分为以下 10 个步骤:

① 购物者浏览商家网站将商品放入购物车, 最后请求购买结算 (此时商家已经获取所有必需的数据, 包括卡号 PAN (Personal Account Number) 和用户设备信息)。

② 商家通过 MPI 将 PAN 和可用的用户设备信息发送到目录服务器。

③ 目录服务器向匹配的 ACS 发出查询请求, 验证 PAN 和设备信息是否合法 (如果这时没有可用的 ACS 进行响应, 则目录服务器会为 MPI 创建一个响应消息并跳到步骤⑤)。

④ ACS 向目录服务器发出响应消息。

⑤ 目录服务器将 ACS 的响应 (或步骤③中自己创建的响应) 回送给 MPI。如果 PAN

(和可用的设备信息)是合法的,那么 3-D 过程就继续进行下去;如果没有证据来证明 PAN (和可用的设备信息)是合法的,则 3-D 处理过程就终止。

⑥ MPI 通过购物者设备(如 PC 浏览器)将支付认证请求送到 ACS, ACS 生成认证消息 Web 页面并发到持卡人浏览器中。

⑦ 持卡人在认证消息框中输入认证信息,并提交给 ACS。

⑧ ACS 利用持卡人输入的信息鉴别购物者的身份,然后 ACS 会生成认证响应结果消息并签名。

⑨ ACS 通过购物者设备将认证响应结果消息返回给 MPI,同时 ACS 将其中特定的数据送到认证历史服务器并记录下来。

⑩ MPI 验证认证响应消息的签名,并将获得授权的交易信息提交给商家的收单行。最后,收单行和发卡行通过它们之间的授权系统(如 VisaNet)进行结算,并将结果返回给商家。

3-D Secure 工作流程示意图如图 20-11 所示。

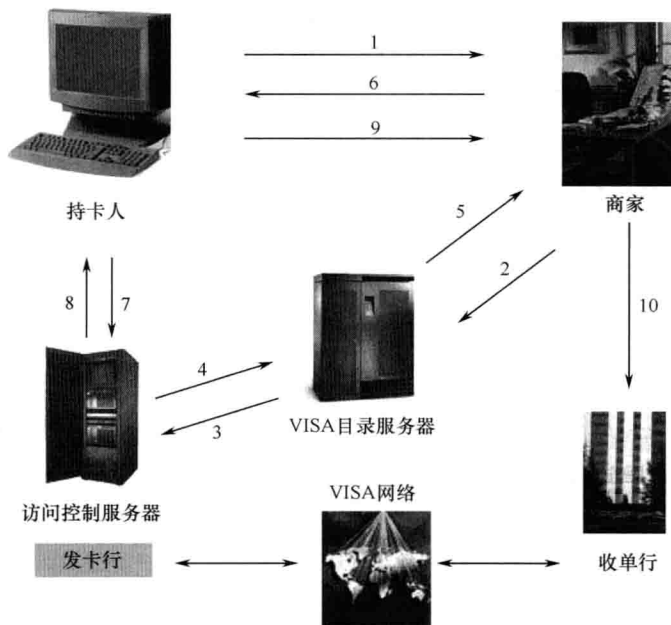


图 20-11 3-D Secure 支付流程示意图

20.7 WAP

WAP (Wireless Application Protocol) 为无线应用协议,是一项全球性的网络通信协议。WAP 使移动互联网有了一个通行的标准,其目标是将互联网中丰富的信息及先进的业务引入到移动电话等无线终端中。WAP 定义了一种通用平台,将互联网上 HTML 描述的信息转换成用 WML (Wireless Markup Language) 描述的信息,直接显示在移动电话的显示屏上。WAP 只要求移动电话和 WAP 代理服务器的支持,而不要求现有的移动通信网络协议做任何的改动,因而可以广泛应用于 GSM、CDMA、TDMA、3G 等多种网络中。

1. 访问模型

(1) WWW 访问模型

WWW 模型提供了易伸缩、功能强大的访问模型，如图 20-12 所示。应用和内容通过标准的格式进行提供，应用端通过浏览器进行浏览。

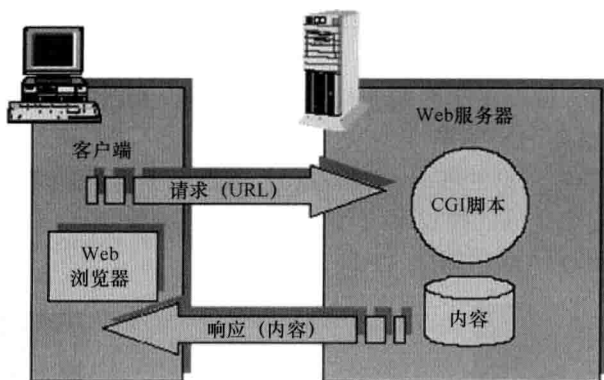


图 20-12 WWW 访问模型

WWW 协议定义了 3 种类型的服务器：

- ① 源服务器：资源驻留和功能创建服务器。
- ② 代理服务器：当客户机对服务器发起请求时，对客户机扮演服务器的角色，对服务器扮演客户机的角色。代理服务器通常处于无法直接通信的客户机和服务器之间，在 WWW 协议中，代理服务器必须既执行服务器又执行客户机的功能。
- ③ 网关：处理不同服务器之间的交换。

(2) WAP 访问模型

WAP 访问模型如图 20-13 所示，与 WWW 访问模型相似，这种相似性为应用开发者提供了极大的便利，包括熟悉的编程模型、已经证明过的结构和利用现有工具的能力（如 Web 服务器，XML 工具等）。为了适应无线环境的应用，已经对其进行了优化和扩展。

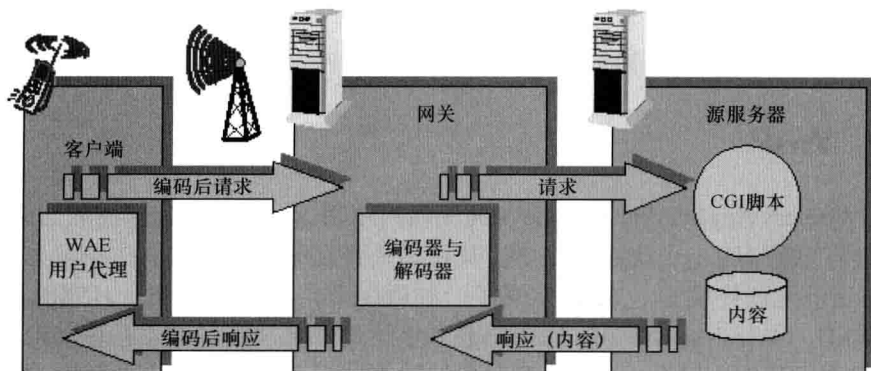


图 20-13 WAP 访问模型

WAP 定义了一组在移动用户终端和网络服务器之间通信的标准组件,包括:

- ① 标准命名模型:完成 WAP 内容的定位。
- ② 内容类型:所有的 WAP 内容以与 WWW 兼容的方式提供。
- ③ 标准内容格式:与 WWW 格式兼容。
- ④ 标准的通信协议:移动设备之间是兼容的。

WAP 的内容和协议是针对手持无线设备市场经过优化的,通过代理技术实现无线领域与 WWW 的连接,WAP 代理具有下列功能:

① 协议网关:完成从 WAP 协议栈(包括 WSP、WTP、WTLS 和 WDP)的请求到 WWW 协议栈(HTTP、TCP/IP)的转换。

② 内容编码和解码:将 WAP 的内容转化为紧凑编码格式以减少网络数据传输量。

通过使用代理技术,移动终端用户可以浏览大量的 WAP 内容,应用开发者也能开发出大量与具体终端无关的应用服务。同时,WAP 代理允许内容和应用驻留在 WWW 服务器上,并且采用成熟的 WWW 技术来开发应用。标准的模型包括 WAP 客户机、WAP 代理以及 WAP 服务器。但 WAP 体系结构可以支持其他的配置。比如,把 WAP 代理的功能包含在 WAP 服务器中,这样就可以实现客户与服务器之间安全的端到端连接。

2. 协议栈模型

WAP 是一个全球性的标准。1997 年 6 月,爱立信、诺基亚、摩托罗拉和 Unwired Planet(即现在的 phone.com)成立了 WAP 论坛,为基于 Internet 的各种服务在移动终端上的应用制定工业标准。同年 11 月,首次公布了该标准的结构。

1998 年 1 月,WAP Forum Ltd 成立,负责监督 WAP 标准的制定。WAP 论坛开始吸收新成员加入,以促进 WAP 在全球无线通信领域的应用和发展。1998 年 5 月,WAP 论坛推出了 WAP 协议的 1.0 版,1999 年 9 月,这一版本被更新的 1.1 版所取代,WAP v1.1 在 WAP v1.0 的基础上,增强了其兼容性和可互操作性,并参考 W3C(万维网联盟)新公布的 XHTML 协议对 WAP 协议的 WML 部分做了修改。随后,WAP v1.2 于 1999 年 12 月发布,增加了 WAP PUSH 结构这部分内容,在 WAP v1.1 的基础上,对 WTA(无线电话应用)部分做了一些补充,并增加了 WAP 可支持的承载网类型。

WAP v2.0 于 2001 年 8 月正式发布,它在 WAP 1.x 的基础上集成了 Internet 上最新的标准和技术,并将这些技术和标准应用到无线领域。这些新技术和标准包括 XHTML、TCP/IP、HTTP/1.1、TLS 等。在这些新技术的支持下,新增加了数据同步、多媒体信息服务、统一存储接口、配置信息提供和小图片等新的业务和应用,同时加强了无线电话应用、Push 技术和用户代理特征描述等原有的应用。这些新的业务和应用将会带来一种全新的使用感受,并极大地激发人们对无线应用服务的兴趣,从而推动移动互联网的发展。与 WAP 1.x 相比,WAP 2.0 协议取消了 WSP、WDP,代之以 HTTP 和 TCP/IP。这种无线数据传输技术的改进带来了传输速率及传输可靠性的有效提高。

WAP v1.x 的协议栈模型如图 20-14 所示。WAP v1.x 安全性依赖于 WTLS 协议,通过 WTLS 实现传输层数据的机密性、完整性和通信双方的身份认证。WAP v1.1 并不提供不可否认的安全保护,即没有实现数字签名,无法做到不可抵赖性。WAP v1.2 通过 WMLSCrypt 提供签名机制来实现不可抵赖性。

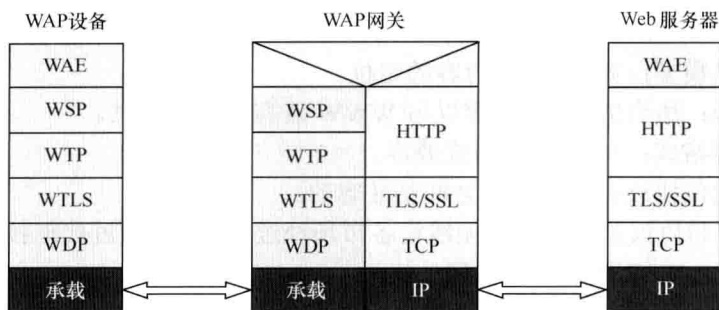


图 20-14 WAP v1.x 协议栈

WAP v2.0 的协议栈模型如图 20-15 所示。WAP v2.0 用 TLS 替换了 WTLS 协议，而且通过 TLS 协议实现了端到端的安全性。

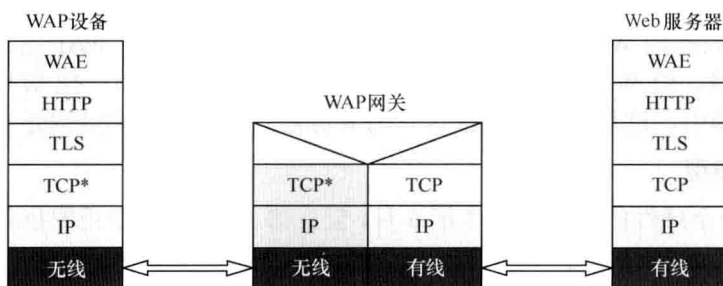


图 20-15 WAP v2.0 协议栈

20.8 S/MIME

1. 简介

S/MIME 是 Secure/Multipurpose Internet Mail Extensions 的缩写，是一种 Internet 标准，在安全方面对 MIME 协议进行了扩展，它可以把 MIME 实体（如数字签名和加密信息等）封装成安全对象，为电子信息应用增添了消息真实性、完整性和保密性服务。S/MIME 不局限于电子邮件，也可以被其他支持 MIME 的传输机制使用，如 HTTP。在 S/MIME 之前，使用最广泛的电子邮件协议为简单邮件传输协议（SMTP，Simple Mail Transfer Protocol），该协议由于其内在的原因而缺乏安全性。

S/MIME v1 是由许多安全供应商于 1995 年开发出来的，只是实现邮件安全性的几个规范之一。例如，Pretty Good Privacy（PGP）是实现邮件安全性的另一个规范。在 S/MIME v1 推出时，安全邮件并没有公认的单一标准，但有几个相互竞争的标准。

1998 年，随着 S/MIME v2 的推出，情况开始发生变化。与版本 1 不同的是，出于希望成为 Internet 标准的考虑，S/MIME v2 被提交到 Internet 工程任务组（IETF）。通过这一步，S/MIME 从许多可能的标准中脱颖而出，从而成为邮件安全标准的领跑者。S/MIME 2 由两份 IETF 征求意见文档（RFC）组成：建立邮件标准的 RFC 2311 和建立证书处理标准的 RFC 2312。这两份 RFC 共同提供了第一个基于 Internet 标准的框架，供应商可以按照该框架来提出可互操作的邮件安全解决方案。有了 S/MIME 2，S/MIME 开始成为邮件安全的

标准。

1999 年,为增强 S/MIME 功能,IETF 提议使用 S/MIME 版本 3。RFC 2632 建立在 RFC 2311 基础上,指定为安全邮件的标准;RFC 2633 增强了证书处理的 RFC 2312 规范。RFC 2634 通过向 S/MIME 添加其他服务扩展了总体功能,如安全回执、三层包装和安全标签。目前,S/MIME v3 已被广泛接受为邮件安全标准,很多邮件软件产品均支持 S/MIME v3,如 Microsoft Outlook、Microsoft Exchange、Foxmail 等。

S/MIME 增加了新的 MIME 数据类型,用于提供数据保密、完整性保护、认证和鉴定服务等功能,这些数据类型包括: application/pkcs7-MIME、multipart/signed 和 application/pkcs7-signature 等。如果邮件包含了上述 MIME 复合数据,邮件中将带有有关的 MIME 附件。在邮件的客户端,接收者在阅读邮件之前,S/MIME 模块将处理这些附件。

S/MIME 只保护邮件的邮件主体,对头部信息则不进行加密,以便让邮件成功地在发送者和接收者的网关之间传递。S/MIME 提供两种安全服务:数字签名和邮件加密。

2. 数字签名

(1) 基本原理

数字签名提供了下列安全功能:

① 身份验证。通过签名来验证身份。它能够将该实体与其他所有实体区分开来,并证明它的唯一性,从而确认“您是谁”这个问题的答案。由于 SMTP 电子邮件中不存在身份验证,因此无法知道实际上是谁发送了邮件。数字签名中的身份验证使收件人可以知道邮件是声称已发送该邮件的那个人或组织发送的,从而解决了这一问题。

② 认可。签名的唯一性可防止签名的所有者否认签名,此功能称为“认可”。因此,签名所提供的身份验证提供了一种强制认可的手段。人们最熟悉的是书面合同上下文中认可的概念:已签名的合同是具有法律约束力的文档,要否认已通过身份验证的签名是不可能的。数字签名提供相同的功能,并且,在某些领域中,逐渐被公认为与书面签名一样具有法律约束力。由于 SMTP 电子邮件不提供身份验证手段,因此无法提供认可功能。发件人很容易否认自己是某个 SMTP 电子邮件的所有者。

③ 数据完整性。数字签名提供的另一安全服务是数据完整性。数据完整性是使数字签名成为可能的特定操作的结果。有了数据完整性服务,当经过数字签名的电子邮件的收件人验证数字签名时,他们可以确信所收到的电子邮件确实是被签名并发送出来的那封邮件,并且在传送过程中未发生改变。如果邮件在签名后的传送过程中发生了任何改变,该签名都将无效。这样,数字签名便能够提供书面签名所无法提供的保证功能,因为书面文档在经过签名后可能被改变。

虽然数字签名提供数据完整性,但不提供保密性。与 SMTP 邮件类似,仅有数字签名的邮件将以明文形式发送,并且可能被其他人阅读。如果邮件是不透明签名的邮件,则会出现一定程度的混乱局面,这是因为虽然邮件是以 Base64 编码的,但它仍然是明文形式。若要保护电子邮件的内容,必须使用邮件加密。

身份验证、认可和数据完整性是数字签名的核心功能。在它们的共同作用下,可以使收件人确信邮件来自发件人,并且所收到的邮件是所发送的邮件。

简单来说,数字签名的工作形式是在邮件发送时对电子邮件的文本执行签名操作,而

在邮件被阅读时执行验证操作，如图 20-16 所示。



图 20-16 S/MIME 数字签名服务

(2) 发送邮件

发送邮件时，执行签名操作所需要的信息只能由发件人提供。在签名操作中，通过捕获电子邮件并对邮件执行签名操作来使用此信息。此操作产生实际的数字签名。然后，此签名将附加到电子邮件中，并随同邮件一起发送。邮件签名过程的具体步骤如下：

- ① 捕获邮件。
- ② 检索用来唯一标识发件人的信息。
- ③ 使用发件人的唯一信息对邮件执行签名操作，以产生数字签名。
- ④ 将数字签名附加到邮件中。
- ⑤ 发送邮件。

图 20-17 显示了邮件签名过程。



图 20-17 S/MIME 邮件签名过程

由于此操作需要来自发件人的唯一信息，因此数字签名提供了身份验证和认可功能。此唯一信息可以证明邮件只能来自该发件人。

(3) 接收邮件

当收件人打开经过数字签名的电子邮件时，系统会对数字签名执行验证过程，并不会从邮件中检索邮件所包含的数字签名。此外还会检索原始邮件，然后执行签名操作，从而产生另一个数字签名。将邮件所包含的数字签名与收件人所产生的数字签名进行比较，如果签名匹配，则证明邮件确实来自所声称的那个发件人；如果签名不匹配，则将邮件标记为无效。邮件签名验证过程的具体步骤如下：

- ① 接收邮件。
- ② 从邮件中检索数字签名。
- ③ 检索邮件。
- ④ 检索用来标识发件人的信息。
- ⑤ 对邮件执行签名操作。
- ⑥ 将邮件所附带的数字签名与收到邮件后所产生的数字签名进行比较。
- ⑦ 如果数字签名匹配，则说明邮件有效。

图 20-18 显示了邮件签名验证过程。



图 20-18 S/MIME 邮件验证过程

验证签名时所使用的发件人信息与对邮件进行签名时发件人所提供的信息不是同一个信息。通过这样一种方式叙述收件人使用的信息：使收件人在验证发件人的唯一信息时不必实际知道该信息，从而保护发件人的信息。

同时采用数字签名过程和数字签名验证过程可验证电子邮件发件人的身份，并确定已签名的邮件中数据的完整性。验证发件人身份会提供其他认可功能，即防止已通过身份验证的发件人声称他们未发送过该邮件。数字签名是防止假冒身份和篡改数据的解决方案，而假冒和篡改这两种情况在基于标准 SMTP 的 Internet 电子邮件中均有可能出现。

3. 邮件加密

(1) 基本原理

邮件加密提供了针对信息泄露的解决方案。基于 SMTP 的 Internet 电子邮件并不确保邮件的安全性。SMTP Internet 电子邮件可能被在发送过程中看到它或在所存储的位置查看它的任何人阅读，S/MIME 已通过采用加密措施解决了这些问题。

加密是一种更改信息的方式，它使信息在重新变为可读或可理解的形式之前无法阅读或理解。虽然邮件加密不像数字签名那样普遍使用，但是它确实解决了被许多人认为是 Internet 电子邮件的最重大缺陷的问题。

邮件加密提供两种特定的安全服务：

① 保密性。邮件加密用来保护电子邮件的内容。只有预期的收件人能够查看该内容，因而该内容是保密的，不会被可能收到或查看到该邮件的其他任何人知道。加密在邮件传送和存储过程中均提供保密性。

② 数据完整性。使用数字签名时，由于采用了使加密成为可能的特定操作，因此邮件加密提供了数据完整性服务。

虽然邮件加密提供保密性，但它不会以任何方式验证邮件发件人的身份。已加密但未签名的邮件与未加密的邮件一样，很容易被他人假冒为发件人。由于认可是身份验证的直接结果，因此邮件加密也不提供认可。虽然加密提供了数据完整性，但是加密的邮件可能仅显示邮件自发送以来未发生过改变，而不提供有关邮件发件人的信息。要证明发件人的身份，邮件必须使用数字签名。

保密性和数据完整性提供了邮件加密的核心功能。它们确保了只有预期的收件人才能查看邮件，并且所收到的邮件就是所发送的邮件。

邮件加密通过在发送邮件时对邮件执行加密操作来使邮件的文本不可读。收到邮件时，通过在阅读邮件时执行解密操作来使文本再次成为可读文本，如图 20-19 所示。



图 20-19 S/MIME 邮件加密服务

(2) 发送邮件

加密操作在发送邮件时执行，它捕获电子邮件，并使用预期收件人所特有的信息来对邮件进行加密。加密的邮件替换了原始邮件，然后将邮件发送至收件人。邮件加密过程的具体步骤如下：

- ① 捕获邮件。
- ② 检索用来唯一标识收件人的信息。
- ③ 使用收件人的信息对邮件执行加密操作，以产生加密的邮件。
- ④ 加密的邮件替换邮件中的文本。
- ⑤ 发送邮件。

图 20-20 显示了电子邮件加密过程。



图 20-20 S/MIME 邮件加密过程

由于此操作需要有关收件人的唯一信息，因此邮件加密提供了保密性。只有预期的收件人具有执行解密操作所需的信息，从而确保了只有预期的收件人能够查看邮件，因为必须首先提供收件人的唯一信息，然后才能查看未加密的邮件。

加密邮件时所使用的收件人信息与解密邮件时收件人所提供的信息不是同一个信息。通过这样一种方式叙述发件人使用的信息：使发件人在使用收件人的唯一信息时不会实际知道该信息，从而保护了收件人的信息。

(3) 接收邮件

当收件人打开加密邮件时，会对加密邮件执行解密操作。此时，将同时检索加密的邮件和收件人的唯一信息。然后，使用收件人的唯一信息对加密邮件执行解密操作。此操作返回未加密的邮件，然后该邮件将显示给收件人。如果邮件在传送过程中发生过改变，解密操作将失败。邮件解密过程的具体步骤如下：

- ① 接收邮件。
- ② 检索加密邮件。
- ③ 检索用来唯一标识收件人的信息。
- ④ 使用收件人的唯一信息对加密邮件执行解密操作，以产生未加密的邮件。
- ⑤ 将未加密的邮件返回给收件人。

图 20-21 显示了解密电子邮件过程。



图 20-21 S/MIME 邮件解密过程

邮件加密和解密过程提供了电子邮件的保密性。此过程解决了 Internet 电子邮件中的重大缺陷：任何人都可以阅读任何邮件。

4. 数字签名与邮件加密

数字签名和邮件加密并不是相互排斥的服务，每个服务都可解决特定的安全问题。数字签名解决身份验证和认可问题，而邮件加密则解决保密性问题。由于每个服务解决不同的问题，因此邮件安全策略通常同时需要这两个服务。这两个服务被设计为一起使用，因为它们分别针对发件人和收件人关系的某一方。数字签名解决了与发件人有关的安全问题，而加密则主要解决与收件人有关的安全问题。

同时使用数字签名和邮件加密时，用户会同时从这两个服务中受益。在邮件中采用这两个服务不会改变其中任何一个服务的处理过程。

（1）发送邮件

邮件签名与加密过程的具体步骤如下：

- ① 捕获邮件。
- ② 检索用来唯一标识发件人的信息。
- ③ 检索用来唯一标识收件人的信息。
- ④ 使用发件人的唯一信息对邮件执行签名操作，以产生数字签名。
- ⑤ 将数字签名附加到邮件中。
- ⑥ 使用收件人的信息对邮件执行加密操作，以产生加密的邮件。
- ⑦ 用加密后的邮件替换原始邮件。
- ⑧ 发送邮件。

图 20-22 显示了对电子邮件进行签名和加密的过程。

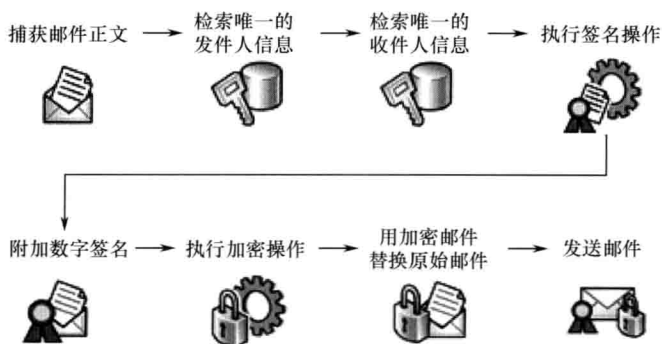


图 20-22 S/MIME 邮件签名与加密过程

(2) 接收邮件

邮件解密和验签过程的具体步骤如下：

- ① 接收邮件。
- ② 检索加密邮件。
- ③ 检索用来唯一标识收件人的信息。
- ④ 使用收件人的唯一信息对加密邮件执行解密操作，以产生未加密的邮件。
- ⑤ 返回未加密的邮件。
- ⑥ 将未加密的邮件返回给收件人。
- ⑦ 从未加密的邮件中检索数字签名。
- ⑧ 检索用来标识发件人的信息。
- ⑨ 使用发件人的信息对未加密的邮件执行签名操作，以产生数字签名。
- ⑩ 将邮件所附带的数字签名与收到邮件后所产生的数字签名进行比较。如果数字签名匹配，则说明邮件有效。

图 20-23 显示了对数字签名进行解密和验证的过程。

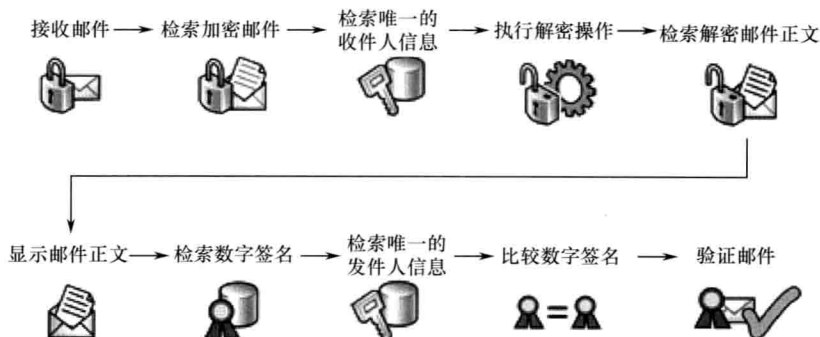


图 20-23 S/MIME 邮件解密与验签过程

5. 三层包装邮件

S/MIME 版本 3 的一个值得注意的增强功能是“三层包装”。三层包装的 S/MIME 邮件是指经过签名、加密、再次签名的邮件。这个额外的加密层为邮件提供了更高层的安全性。

第21章 常见应用

21.1 防止假网站与 Web 服务器证书

本节以网上银行为例来说明如何防止假网站诈骗。

21.1.1 假网站

1. 什么是假网站诈骗

假网站诈骗是“网络钓鱼”的一种，通常是指不法嫌疑人未经许可，以某家银行的名义，通过互联网建立貌似银行网站或网上银行的假网页，并借此发布虚假消息，搜集客户资料，骗取客户网上银行注册卡号（登录 ID）、密码、口令等信息，进而达到非法窃取客户资金的目的。

假网站这种欺诈手法最初出现在北美，2003 年开始在亚洲蔓延，并相继在中国香港地区发生了多宗利用假网站骗取客户网上银行账户口令和资金的案例。从 2004 年开始，假冒银行网站开始在中国大陆出现，网络安全事件也频繁见诸于媒体，使网上银行安全性问题成为公众关注的焦点。仅针对工商银行网银的诈骗假网站就多达几百个。

2. 假网站的表现形式

① 假网站的网址与真网站网址较为接近。由于注册域名的成本非常低，不法分子为增强假网站的欺骗性，往往使用和真实网站网址非常相似的域名。

② 假网站的页面形式和内容与真网站较为相似。假冒网站的页面往往使用正规网站的 LOGO、图表、新闻内容和链接，而且在布局和内容上与真实网站非常相似。

3. 不法分子欺诈的通常手法

(1) 通过病毒传播假网站信息

不法分子克隆一个与银行网站一模一样的网页，并且使用的登录地址也与银行网站的地址非常接近，然后使用一些电脑病毒程序、垃圾软件等将假网站地址发送到客户的电脑上，或放在搜索网站上诱骗客户登录，以窃取客户卡号、密码等信息。如工行曾经发现的“<http://www.lcbc.com.cn>”、“<http://mybank.iclc.com.cn/>”、“<http://www.icbc.dizhen.com>”等假网站，与工行网站地址 <http://www.icbc.com.cn>、<https://mybank.icbc.com.cn> 十分相似。

(2) 通过手机短信，冒充银行名义发送诈骗短信

不法分子利用手机短信，冒充银行名义向客户发送诈骗短信，声称客户中奖或账户被他人盗用等，要求客户尽快登录到短信中指定的网站进行身份验证。而该网站是不法分子建立的、用于套取客户信息的假网站，如果客户登录该网站并进行操作，客户的卡号、密码、身份证件等信息将会被不法分子获悉。

(3) 冒充银行邮箱，发送虚假信息引诱客户登录假网站

不法分子以垃圾邮件的形式大量发送欺诈性邮件，这些邮件多以中奖、顾问、对账等内容，或是以银行账号被冻结、银行系统升级等各种理由，要求收件人点击邮件上的链接地址，登录一个酷似银行网页的界面，而用户一旦在这个指定的登录界面输入了自己的卡（账）号、密码等，这些信息就会被窃取。近期发现，不法分子以所谓“中国工商银行网络安全科”的名义，向客户发送电子邮件，谎称持卡人账户被冻结，并要求客户到指定的网页修改密码。

(4) 建立假电子商务网站，通过假的支付页面窃取客户网上银行信息

不法分子首先建立一个假的电子商务网站，然后在电子商务网站（如淘宝网、腾讯网等）发布虚假的商品信息，该信息中的商品价格往往比市场同类商品便宜很多，同时不法分子还会留下自己的 QQ 号或者 MSN 等即时通信工具号码以及假电子商务网站的网址。当客户对该网站销售的便宜商品动心，并通过该网站购物进行支付时，就会链接到一个假的银行支付页面，客户在假支付页面输入的卡号、密码等信息就会被不法分子获取。

21.1.2 使用 Web 服务器证书预防假网站

1. 网站申请 Web 服务器证书

(1) 生成证书请求文件

不同 Web 服务器生成证书请求的方式不同。

针对支持 JSP 的 Web 服务器，如 Tomcat、WebSphere、WebLogic 等，需要使用 JDK 自带的工具 keytool 生成证书请求文件。keytool -genkey 产生密钥对。keytool -certreq 产生证书请求文件。

针对 IIS 系统，使用 Microsoft 提供的管理工具，按照 IIS 证书向导的操作提示即可生成证书请求文件。

证书请求文件内容的格式遵循 PKCS#10 规范，其中包含网站基本信息和网站 RSA 公钥，如下例所示：

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIDSDCCArECAQAwbTESMBAGA1UEAxMJbG9jYWxob3N0MRUwEwYDVQQLHgxibiVfo
i6SLwU4tX8MxETAPBgNVBAoeCFMXedFmelNaMQ8wDQYDVQQHHGZtd23AUzoxDzAN
BgNVBAgeBIMXTqxeAjELMAkGA1UEBhMCQ04wgZ8wDQYJKoZIhvcNAQEBBQADgY0A
MIGJAoGBAMFbTnn0+hOmJc+jSV3FJDpe0seH2ojbzESzEudIoMxNfStIg3Vi0gyx
0WP1JYbnXMB1g6q/ZAUOtU0nCCs9hVqFj2irrm66GN1UjhqrcWsJjbgNSAf2C3Nx
O0ygc91/DI0LHXhcs5p3p5cv6bTc0wEh7G3wGYH7ZG/PM3CDcs3AgMBAAAGggGZ
MBoGCisGAQQBgicNAgMxDBYKNS4xLjI2MDAuMjB7BgorBgEEAYI3AgEOMW0wazAO
BgNVHQ8BAf8EBAMCBPAwRAYJKoZIhvcNAQkPBDCwNTAOBggqhkiG9w0DAgICAIAw
DgYIKoZIhvcNAwQCAgCAMAcGBSsOAwhMAAoGCCqGSIb3DQMHMBMGA1UdJQQMMAoG
CCsGAQUFBwMBMIH9BgorBgEEAYI3DQICMYHuMIHrAgEBHloATQBpAGMAcGvAHMA
bwBmAHQAIABSAFMAQQAgAFMAQwBoAGEAbgBuAGUAbAAgAEMAcbB5AHAAdABvAGcA
cgBhAHAAaABpAGMAIABQAHIAbwB2AGkAZABIAHIDgYkAbII1TrHis4afw+wbLrZI
OYe1boagX3QNYHNj4kpktRyBgIFt6WofQ1nXK6TXmpAm2/AmY20/h+a1GZ1/vn7E
EzHcNQfjvHoSZH7yU5FzvBV5sPGGZ//dlrLYX0iY8qhQicTdPQT3MRoYjUKBvi7I
```



```

RJnfbWbpQKIZSwebIEKNIIYAAAAAAAAAADANBgkqhkiG9w0BAQUFAAOBgQCtkhCM
XPmXwVw2TXBSLHXmj+21LD4VPF5pi/rg+itp3iFxQQ9A7hmaH8ICIFlBjx6dQUda
De1fNCa34E1nOfP3epZQAGqdyO2ulQ9CsTkmi+bP705tj9t2zU6G7gfYh+qvWnO4
ByupOjtwjLPzAlRBPX7SRPaKcgnlH+YMAJVI5Q==
-----END NEW CERTIFICATE REQUEST-----

```

(2) 签发证书

将证书请求文件内容提交给 CA 系统，CA 系统解析该 PKCS#10 请求包，获得网站基本信息 and 网站 RSA 公钥，然后使用 CA 私钥为该网站签发 Web 服务器证书。

(3) 安装证书

不同 Web 服务器安装证书的方式不同。

针对支持 JSP 的 Web 服务器，如 Tomcat、WebSphere、WebLogic 等，需要使用 JDK 自带的工具 keytool 安装证书。用 keytool -import 导入证书。

针对 IIS 系统，使用 Microsoft 提供的管理工具，按照 IIS 证书向导的操作提示即可完成证书安装。

(4) 启用 SSL

不同 Web 服务器启用 SSL 的方式不同。

有些 Web 服务器需要手工修改配置文件来配置并启用 SSL。如 Tomcat，需要修改 server.xml 文件。有些 Web 服务器使用管理工具配置并启用 SSL。如 IIS，使用 Microsoft 提供的管理工具。

2. 用户鉴别真假网站

(1) 核对网站域名

假网站的网址与真网站网址较为接近，需要自己辨别其不同之处。如假冒网站通常将英文字母 I 替换为数字 1，CCTV 被换成 CCYV 或者 CCTV-VIP 这样的仿造域名。

(2) 关注浏览器提示

当用户访问安装 Web 服务器证书的网站时，用户浏览器与网站之间将建立 SSL 加密通道。鉴于数字证书技术的复杂性，为降低该技术使用的复杂性和提高用户使用的方便性，浏览器中已经预装多个可信的 CA 证书，同时浏览器将帮助用户对 Web 服务器证书的有效性进行自动验证。

当发现如下疑问或异常时，浏览器将向用户进行提示，部分情况将由用户选择是否继续进行：

- ① 浏览器没有找到可信的 CA 证书来验证 Web 证书中数字签名是否正确。
- ② Web 证书中起始日期 NotBefore 在当前日期之后，或终止日期 NotAfter 在当前日期之前。
- ③ Web 证书中申请者的 CN 项与当前访问的网站域名或 IP 不一致。
- ④ Web 证书中密钥用途 KeyUsage、ExtKeyUsage 不正确。

(3) 核对安全证书

用户浏览器与网站建立 SSL 加密通道成功后，IE 浏览器右下角的状态栏上会显示一个“挂锁”图形的安全证书标识。单击挂锁，将显示该网站证书的内容，如图 21-1 所示。

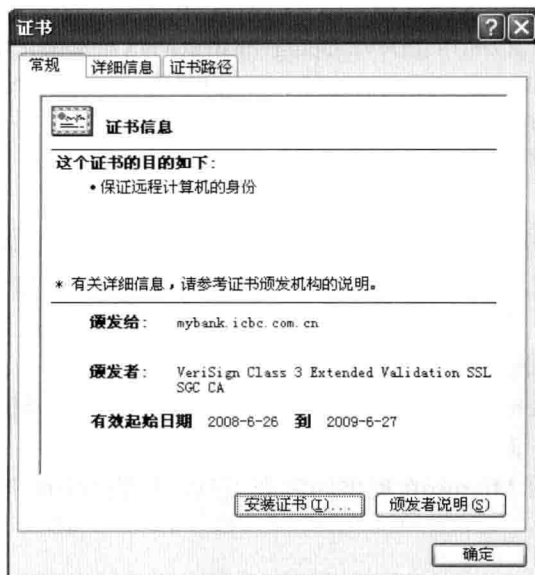


图 21-1 网站证书的内容

用户需要仔细核对该证书中网站信息是否正确，如发现疑问或异常，则该网站可能是假网站。网站信息主要包括：国家 C、省份 S、城市 L、单位 U、部门 OU、名称 CN 等。

21.2 防止假软件与代码签名证书

本节以网上银行为例来说明如何防止假软件。

21.2.1 Web 技术的发展

由于 Web 技术具有跨平台和瘦客户端等技术优势，因此网上银行及电子交易类系统普遍采用 Web 技术，既简化了用户的操作难度，又降低了系统的运维成本。

Web 技术主要由传输协议、服务器端技术和浏览器端技术组成。

最早的 Web 技术起源于静态页面的浏览，传输协议采用 HTTP，服务器端只负责存储 HTML 静态页面，浏览器端只负责显示 HTML 静态页面。为保证浏览器端计算机的安全性，浏览器端在操作系统环境中构建封闭的、安全隔离的动态运行环境，不允许访问任何本地软硬件资源（如文件、磁盘等）。

由于基于静态页面浏览的 Web 技术存在很多局限性，为适应复杂多变的 Web 应用需求，Web 技术自诞生至今的十几年内得到了快速发展。

1. 传输协议

传输协议已由最初的 HTTP/1.0 发展到目前的 HTTP/1.1。

HTTP 协议是基于请求 / 响应模式的。浏览器与服务器建立连接后，发送一个请求给服务器。服务器接到请求后，进行相应处理并返回对应的响应信息。浏览器收到响应后，进行处理并显示。

HTTP/1.1 相较于 HTTP/1.0 主要区别有：缓存处理、带宽优化及网络连接的使用、错误通知的管理、消息在网络中的发送、互联网地址的维护、安全性及完整性等。

2. 服务器端技术

服务器端技术的发展目的是静态向动态发展，主要包括 CGI、PHP、ASP、ASP.NET、Servlet 和 JSP 技术。

CGI (Common Gateway Interface) 技术，即公共网关接口技术。最早的 Web 服务器简单地响应浏览器发来的 HTTP 请求，并将存储在服务器上的 HTML 文件返回给浏览器。CGI 是第一种使服务器能根据运行时的具体情况，动态生成 HTML 页面的技术。1993 年，NCSA (National Center for Supercomputing Applications) 提出 CGI1.0 的标准草案，之后分别在 1995 年和 1997 年制定了 CGI 1.1 和 1.2 标准。CGI 技术允许服务段的应用程序根据客户端的请求动态生成 HTML 页面，这使客户端和服务端端的动态信息交换成为了可能。随着 CGI 技术的普及，聊天室、论坛、电子商务、信息查询、全文检索等各式各样的 Web 应用蓬勃兴起，人们可以享受到信息检索、信息交换、信息处理等各种更为便捷的信息服务了。

PHP (Personal Home Page Tools) 技术是 1994 年由 Rasmus Lerdorf 发明专用于 Web 服务端编程的 PHP 语言。与以往的 CGI 程序不同，PHP 语言将 HTML 代码和 PHP 指令合成为完整的服务端动态页面，Web 应用的开发者可以用一种更加简便、快捷的方式实现动态 Web 功能。

ASP (Active Server Pages) 技术即活动服务器页面技术。1996 年，Microsoft 借鉴 PHP 的思想，在其 Web 服务器 IIS 3.0 中引入了 ASP 技术。ASP 使用的脚本语言是我们熟悉的 VBScript 和 Javascript。借助 Microsoft Visual Studio 等开发工具在市场上的成功，ASP 迅速成为 Windows 系统下 Web 服务端的主流开发技术。

ASP.NET 技术是面向下一代企业级网络计算的 Web 平台，是对传统 ASP 技术的重大升级和更新。ASP.NET 是建立 .NET Framework 的公共语言运行库上的编程框架，可用于在服务器上生成功能强大的 Web 应用程序。

Servlet 与 JSP 技术由以 Sun 公司为首的 Java 阵营于 1997 年和 1998 年分别推出。JSP 与 Servlet 的组合让 Java 开发者同时拥有了类似 CGI 程序的集中处理功能和类似 PHP 的 HTML 嵌入功能。此外，Java 的运行时编译技术也大大提高了 Servlet 和 JSP 的执行效率。Servlet 和 JSP 被后来的 J2EE 平台吸纳为核心技术。

3. 浏览器端技术

浏览器端技术发展的目的是从静态向动态发展、从无权访问本地资源向有权访问发展，主要包括 HTML 语言、Java Applets、脚本程序、CSS、DHTML、插件技术等。其中，插件技术主要解决本地资源访问的问题。

HTML 是 Hypertext Markup Language (超文本标记语言) 的缩写，它是构成 Web 页面的主要工具。

Java Applet，即 Java 小应用程序。使用 Java 语言创建小应用程序，浏览器可以将 Java Applets 从服务器下载到浏览器，在浏览器所在的机器上运行。Java Applets 可提供动画、音频和音乐等多媒体服务。1996 年，著名的 Netscape 浏览器在其 2.0 版本中率先提供了对

Java Applets 的支持,随后,Microsoft 的 IE 3.0 也在这一年开始支持 Java 技术。Java Applets 使得 Web 页面从只能展现静态的文本或图像信息,发展到可以动态展现丰富多样的信息。动态 Web 页面不仅仅表现在网页的视觉展示方式上,更重要的是它可以对网页中的内容进行控制与修改。

脚本程序是嵌入在 HTML 文档中的程序。使用脚本程序可以创建动态页面,从而大大提高了交互性。用于编写脚本程序的语言主要有 JavaScript 和 VBScript。JavaScript 由 Netscape 公司开发,具有易于使用、变量类型灵活和无须编译等特点。VBScript 由 Microsoft 公司开发,与 JavaScript 一样,可用于设计交互的 Web 页面。需要说明的是,虽然 JavaScript 和 VBScript 语言最初都是为创建客户端动态页面而设计的,但它们都可以用于服务端脚本程序的编写。客户端脚本与服务端脚本程序的区别在于执行的位置不同,前者在客户端机器执行,而后者是在 Web 服务端机器执行。

CSS (Cascading Style Sheets),即级联样式表。1996 年底,W3C 提出了 CSS 的建议标准,同年,IE 3.0 引入了对 CSS 的支持。CSS 大大提高了开发者对信息展现格式的控制能力,1997 年的 Netscape 4.0 不但支持 CSS,而且增加了许多 Netscape 公司自定义的动态 HTML 标记,这些标记在 CSS 的基础上让 HTML 页面中的各种要素“活动”了起来。

DHTML (Dynamic HTML),即动态 HTML。1997 年,微软发布了 IE 4.0,并将动态 HTML 标记、CSS 和动态对象 (Dynamic Object Model) 发展成为一套完整、实用、高效的客户端开发技术体系,微软称其为 DHTML。同样是实现 HTML 页面的动态效果,DHTML 技术无须启动 Java 虚拟机或其他脚本环境,可以在浏览器的支持下,获得更好的展现效果和更高的执行效率。

插件技术大大丰富了浏览器的多媒体信息展示功能,常见的插件包括 QuickTime、Realplayer、Media Player 和 Flash 等。为了在 HTML 页面中实现音频、视频等更为复杂的多媒体应用,1996 年的 Netscape 2.0 成功地引入了对 QuickTime 插件的支持,插件这种开发方式也迅速风靡了浏览器的世界。同年,在 Windows 平台上,微软将 COM 和 ActiveX 技术应用于 IE 浏览器中,其推出的 IE 3.0 正式支持在 HTML 页面中插入 ActiveX 控件,这为其他厂商扩展 Web 客户端的信息展现方式提供了方便的途径。1999 年,Realplayer 插件先后在 Netscape 和 IE 浏览器中取得了成功,与此同时,微软自己的媒体播放插件 Media Player 也被预装到了各种 Windows 版本之中。同样具有重要意义的还有 Flash 插件的问世:20 世纪 90 年代初期,Jonathan Gay 在 FutureWave 公司开发了一种名为 Future Splash Animator 的二维矢量动画展示工具,1996 年,Macromedia 公司收购了 FutureWave,并将 Jonathan Gayde 的发明改名为我们熟悉的 Flash。从此,Flash 动画成了 Web 开发者表现自我、展示个性的最佳方式。

21.2.2 插件技术与假网银软件

插件技术主要用于解决浏览器对本地资源的访问问题。网上银行和电子交易网站采用数字证书技术来保证其安全性,且用户的私钥及数字证书存储在本机密码模块中,属于本地资源。

为了在网上银行或电子交易业务操作过程中访问用户的密码模块,需要在网站中增加插件形式的网银软件或交易软件,通过该软件来实现对用户密码模块的各种密码操作和数

据访问,因此该软件的安全性直接影响到网上银行或电子交易系统的安全性。

由于 Web 技术是开放的,且网上银行和电子交易直接与资金有关,因此该类网站逐渐成为不法分子的攻击对象,针对该类网站的木马程序显著增加,这种木马程序伪装成网银软件或交易软件,对用户本地密码模块进行偷窃、劫持等,导致用户资金被非法转移。

21.2.3 使用代码签名证书预防假网银软件

代码签名证书为软件开发商提供了一个理想的解决方案,使得软件开发商能对其软件代码进行数字签名。通过对代码的数字签名来标识软件来源以及软件开发者的真实身份,保证代码在签名之后不被恶意篡改。使用户在下载已经签名的代码时能够有效地验证该代码的可信度。

从用户角度,可以通过代码签名服务鉴别软件的发布者及软件在传输过程中是否被篡改。如果某软件在用户计算机上执行后造成恶性后果,由于代码签名服务的可审计性,用户可依法向软件发布者索取赔偿,此举可有效制止软件开发者发布攻击性代码的行为。

从软件开发者和 Web 管理者的角度,利用代码签名的抗伪造性,可为其商标和产品建立一定的信誉。利用可信代码服务,一方面开发者可借助代码签名获取更高级别权限的 API,设计各种功能强大的控件和桌面应用程序来创建出丰富多彩的页面;另一方面用户也可以理性地选择所需下载的软件包。并且利用代码签名技术,还可以大大减少客户端防护软件误报病毒或恶意程序的可能性,使用户在多次成功下载并运行具有代码签名的软件后,和开发者间的信任关系得到巩固。

1. 申请代码签名证书

直接向 CA 系统申请代码签名证书。

2. 对网银软件进行代码签名

使用微软提供的代码签名工具 signcode.exe 对网银软件进行代码签名。

3. 用户鉴别真假网银软件

当用户下载网银软件控件时,系统会提示控件是否具有合法的代码签名,如图 21-2 所示。

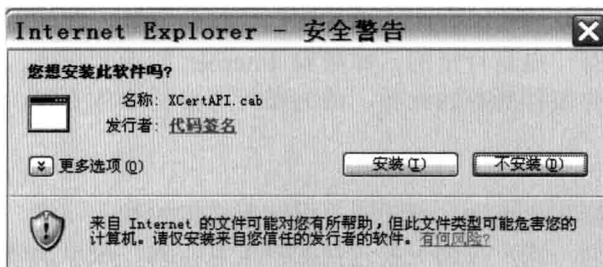


图 21-2 控件下载安全警告

用户可以单击“代码签名”，查看该控件是由谁发布的，如图 21-3 所示。

单击“查看证书”按钮，可以查看发行者的代码签名证书的具体内容，如图 21-4 所示。



图 21-3 代码签名信息

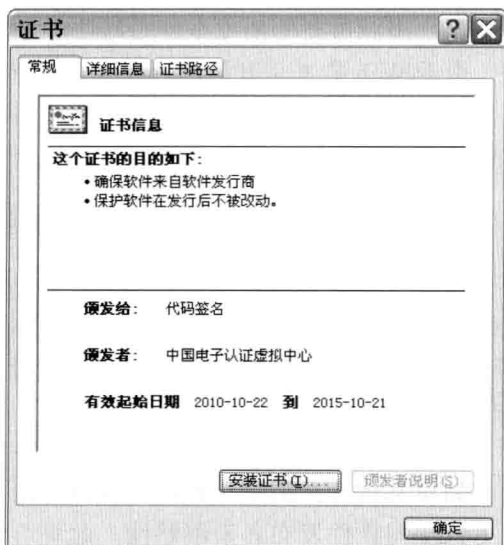


图 21-4 代码签名证书

若该控件的发行者是可信的，如商业银行等，则单击“安装”按钮即可。当出现以下情况时，则该控件可疑，建议用户不要下载：

- ① 该控件没有代码签名。
- ② 该控件的代码签名证书无效。
- ③ 该控件的代码签名证书内容可疑或未知。

21.3 网上银行系统

21.3.1 简介

网上银行是商业银行基于互联网为客户提供安全、实时的银行业务服务。作为一种全新的银行客户服务提交渠道，客户可以不必亲身去银行办理业务，只要能够上网，无论在家里、办公室，还是在旅途中，都能够每天 24 小时安全便捷地管理自己的资产，或者办理查询、转账、缴费等银行业务。

目前国际金融界的发展状况表明，尽管不同的银行有其不同的发展战略，目前正处在不同的发展阶段，但有一点是肯定的，即随着 Internet 的不断发展，随着金融业的不断创新，网上银行必将包含银行所有的业务，成为银行主要的业务手段。网上银行的业务主要有以下几类。

(1) 查询类

查询类业务主要指账户查询。对于网上银行所实现的基本功能，各家银行对其所归置的类别和层级有所不同。账户查询主要包括账户信息查询、账户余额查询以及账户交易明细查询。出于用户对于资金全面的了解和掌控，各家银行业已将资产负债一览作为基本功能，向用户呈现活期、定期、基金、股票、贷款、信用卡等的大概情况。

(2) 交易类

交易类业务主要分为转账汇款、网上支付、自助缴费几类。转账汇款基本功能包括同

城、异地、本行、跨行的资金转账,辅助功能还包括收款人账户管理、付款用途维护、对外转账限额设置等。延伸功能包括预约转账、批量转账,而预约转账又可分为指定日期转账、预约周期转账、指定条件转账。网上支付即网上即时付款服务。网上支付功能在网银中应用得比较普遍,网上支付是为用户提供在网上购买商品的一种渠道。网上支付有多种方式:从网银链接到网上商城、根据网上支付订单号进行支付、从他行的网站链接到网银的支付平台或在网银外进行网上支付等。自助缴费功能为用户提供向各类银行特约收费单位自行缴纳各类日常费用的服务功能,如公共事业性收费、保险续期缴费等。

(3) 投资理财类

投资理财类业务主要包括卡储蓄业务、自助贷款、外汇、黄金、证券、基金、国债、保险、信用卡服务等。卡储蓄业务功能主要体现在个人或家庭可以对所属各类银行卡或存折进行管理、查询和交易,主要指卡内活期转整存整取、零存整取、通知存款、存本取息等。自助贷款业务主要指各种贷款的网上申请、网上放贷、申请展期、贷款试算还贷及额度查询等,可以方便个人或家庭在网上及时办理各类贷款业务,主要包括个人消费贷款、自助质押贷款、住房贷款、汽车贷款等。外汇买卖功能是个人客户委托银行把一种可自由兑换的外币兑换成另一种可自由兑换的外币,并参照国际金融市场行情制定相应汇率。黄金交易分为纸黄金交易和代理实物黄金交易。证券即是早期的网上银行股票买卖交易功能。用户可以在网上银行办理股票委托交易,管理资金账户、查询最新证券信息和股票行情。但由于银监会的监管,目前所有银行已经取消了原来的证券交易功能,只支持银证转账和第三方存管,基本上就是一个银行和证券账户之间的资金互转。基金交易指网上办理与基金相关的各种业务交易,如开户、销户、查询、购买、赎回等。保险主要指网上直接购买保险产品,其有多种方式:直接在网银内选择保险产品并投保、链接到保险公司进行投保或只进行保费的续缴等。国债主要指国债买卖、管理国债账户、查询国债价格和交易明细等。信用卡服务主要指信用卡信息管理、还款设置、网上还款等。

(4) 服务管理类

服务管理类主要包括个人设置、网银设置、财务管理等。个人设置包括信息修改、密码设置、通知提醒等。网银设置为个人用户使用网上银行各项业务功能实时提供在线通知提醒或帮助、说明以及各项用户自助服务功能。财务管理主要是指网上银行提供给客户用于各账户情况的财务管理与分析。

21.3.2 应用安全需求

网上银行以 Internet 等开放式网络环境传输交易数据,而且涉及用户资金转移等敏感信息,所以在用户的身份认证、资金的秘密传输以及数据的完整性方面存在许多安全问题。网上银行服务提供者首先需要确定自己的系统不会受到网络黑客的入侵,造成秘密信息泄露,业务损失或服务中断。对用户而言,必须确认在网络上输入的秘密信息不会被盗用,输入的交易资料不会被篡改并且能正确迅速地传送到接收端系统。

网上银行系统应用安全方面的具体需求主要包括以下内容。

(1) 正确鉴别用户的个人身份及权限

保证用户身份的真实性和合法性、用户权限的有效性。

(2) 保证交易数据的真实性和完整性

防止非法用户对数据进行假冒、篡改和删除,防止数据传送过程中信息的丢失和重复,保证信息传送次序的统一。

(3) 保证交易数据的机密性

通过对一些敏感数据进行加密来保护系统之间的数据交换,防止除接收方之外的第三方截获数据。

(4) 抗抵赖

防止发送消息者事后否认其所发送的消息。

(5) 审计能力

根据机密性和完整性的要求,对交易结果进行记录。

(6) 密钥管理

管理用户在网上银行系统中所使用的密钥,保证密钥的真实性、有效性和完整性。

21.3.3 应用安全总体架构

鉴于网上银行采用开放性的 Web 技术和互联网技术,为全面解决网上银行应用安全需求,通常采用 PKI 体系。基于数字证书技术,网上银行能有效解决用户身份认证、敏感数据保密性、交易数据完整性和交易操作不可抵赖性问题,极大地方便了银行企业客户和个人客户。事实上,数字证书(俗称 U 盾)已经成为国内网上银行的标准配置。如果没有数字证书,企业用户将不允许使用网上银行。上亿个人用户已经通过数字证书访问网上银行实现转账或汇款等资金操作。

网上银行应用安全总体框架如图 21-5 所示。

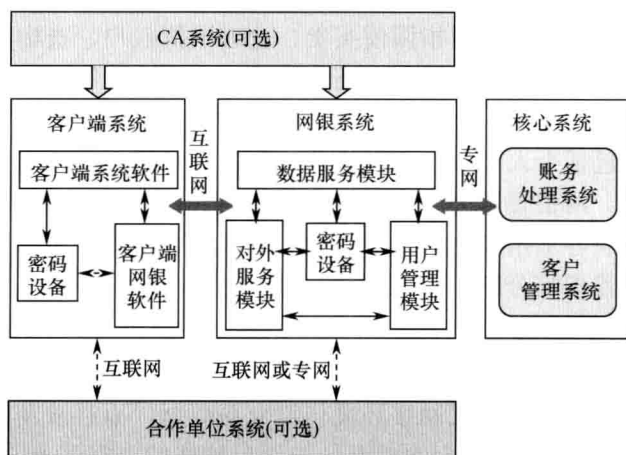


图 21-5 网上银行应用安全总体框架

(1) 关联方

网上银行业务涉及的关联方主要包括 4 类:

- ① 网银用户: 即个人或单位, 通过网银方式获取各种银行业务服务。
- ② 商业银行: 通过网银方式向个人或单位提供各种银行业务服务。
- ③ 合作单位: 依托网银方式对个人或单位提供网上购物、网上缴费、网上充值等非银

行业务服务。

④ CA 机构：给网银业务涉及的上述 3 个关联方颁发数字证书，提供电子认证服务。当不采用第三方电子认证服务时，CA 机构属于商业银行。

（2）系统构成

网上银行整体系统主要包括 4 部分：

① 客户端系统：指基于互联网技术的各种终端系统，包括计算机、智能终端等。主要由密码设备、客户端系统软件和客户端网银软件组成。网银用户通过客户端系统访问网银系统和合作单位系统（有些网银业务不需要访问合作单位系统）。客户端系统通过互联网与网银系统和合作单位系统互联。

② 网银系统：实现网银业务服务、网银客户管理、网银安全保障等功能。主要由对外服务模块、用户管理模块和数据服务模块组成。网银系统通过专网同银行核心系统互联，通过互联网或专网同合作单位系统互联。

③ 合作单位系统：可选，部分业务不包括此系统。

④ 核心系统：实现银行内核心账务处理和客户信息管理。

⑤ CA 系统：实现数字证书的生命周期管理。

客户端系统主要包括：

① 密码设备：实现用户密钥安全存储和密码安全运算。主要包括：USB-Key、IC 卡、动态令牌等。

② 客户端系统软件：管理本地资源，并保证密码设备与客户端网银软件能通信。主要包括：操作系统、密码设备驱动等。

③ 客户端网银软件：访问密码设备和网银系统。对于 B/S 方式主要包括：浏览器、控件/Applet/插件、客户端软件（可选）等。

网银系统主要包括：

① 对外服务模块：实现网银业务服务；对于 B/S 方式，包括 Web 服务模块和应用服务模块。

② 用户管理模块：实现用户信息管理和身份认证等安全保障功能。

③ 数据服务模块：实现数据存储功能。

④ 密码设备：实现网银系统密钥安全存储和密码安全运算。主要包括：加密机、加密卡、SSL 加速器等。

21.4 网上报税系统

21.4.1 简介

随着 IT 技术的迅速发展及其在税收领域的广泛应用，网上税收已成为不可逆转的发展趋势。目前国内多数的税务机关都面向纳税人开展了网上申报业务，并逐步实现了网上实时缴款，极大方便了纳税人，提高了税收征管的效能。

由于网上纳税申报业务依托互联网开展，而互联网固有的开放性又具有用户真实身份验证困难、信息在网络传输过程中保密性差、容易遭受恶意篡改、用户容易抵赖其网络行为等特点。因此，随着系统不断地推广运行，安全问题逐渐显露。

21.4.2 应用安全需求

解决网上申报业务中的安全和责任问题，使无纸化真正落在实处的核心是：电子申报数据应与上门申报的纸质申报表具有同等的法律效力。

网上申报业务应用安全需求分析如下：

- ① 确保纳税人登录网上申报系统身份的可靠性。
- ② 确保纳税人网上申报过程中电子申报数据的保密性和完整性。
- ③ 采取可靠的技术确保网上申报数据的法律效力，取消纳税人网上申报后递交纸质申报材料。
- ④ 纳税人自助打印电子缴款凭证。

21.4.3 应用安全总体架构

网上报税应用安全总体架构如图 21-6 所示。

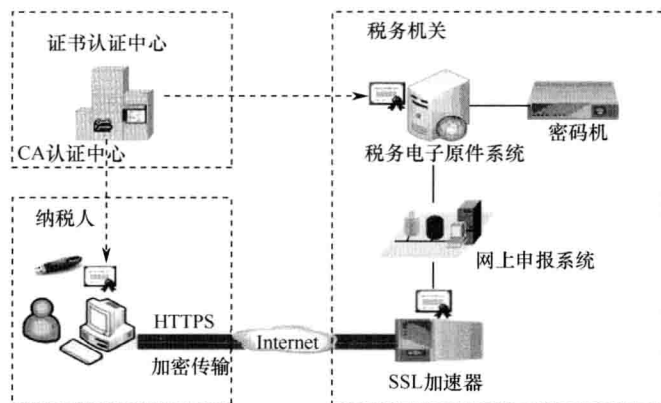


图 21-6 网上报税应用安全总体框架

基于数字证书实现身份认证、利用数字证书实现网上申报数据的可靠电子签名、通过税务电子原件系统对申报电子原件进行可靠管理，使网上申报系统产生的申报电子原件符合电子签名法的相关要求，为税务机关网上申报系统提供具有法律效力的数据电文，为纳税人提供更安全的网上申报服务。

网上报税系统主要包括以下组成部分。

- ① 数字证书：第三方电子认证服务机构为纳税人和税务机关颁发的电子身份凭证，能可靠标识纳税人的网络身份。
- ② 税务电子原件系统：根据纳税人的电子申报数据产生可靠的电子签名，向纳税人提供具有税务机关可靠电子签名的申报回执服务和具有可靠电子签名的申报原件查询下载服务，以及电子缴款凭证自助打印服务。
- ③ SSL 加速器：SSL 加速器是部署在网上申报系统服务器端的硬件，用于提高 SSL 传输处理速度，减轻网上申报系统服务器的负担。
- ④ 数字证书登录系统是一种优于原有用户名密码登录的安全登录方式。

⑤ 电子申报数据中增加了纳税人的电子签名,使申报原件具有了与纸质申报材料同等的效力。

⑥ 税务机关为纳税人提供了申报电子回执,使电子回执与纸质付款凭证具有了同等的法律效力。

⑦ 税务机关为纳税人提供了基于电子回执的、在线自助打印电子缴款凭证服务,使纳税人无须出门即可获取回执并用作记账凭证。

⑧ 纳税人的申报表及相应的电子签名可直接归档处理,实现了申报原件的在线查询及验证。

电子签名法赋予数字签名法律效力后,数字证书技术就可有效解决网上报税中的应用安全问题,使得企业网上报税成为现实,同时提高了企业和税务部门的工作效率。事实上,国内大部分省份的上千万家企业已经通过数字证书在进行网上报税。

21.5 电子病历系统

21.5.1 简介

病历是病人在医院诊断治疗全过程的原始记录,它包含首页、病程记录、检查检验结果、医嘱、手术记录、护理记录等。电子病历不仅指静态病历信息,还包括提供的相关服务。是以电子化方式管理的有关个人终生健康状态和医疗保健行为的信息,涉及病人信息的采集、存储、传输、处理和利用的所有过程信息。美国国立医学研究所将其定义为:电子病例(EMR)是基于一个特定系统的电子化病人记录,该系统提供用户访问完整准确的数据、警示、提示和临床决策支持系统的能力。

电子病历是随着医院计算机管理网络化、信息存储介质(如光盘和IC卡)等的应用及Internet的全球化而产生的。电子病历是信息技术和网络技术在医疗领域的必然产物,是医院病历现代化管理的必然趋势,其在临床的初步应用极大地提高了医院的工作效率和医疗质量,但这还仅仅是电子病历应用的起步。

电子病历(EMR, Electronic Medical Record)也叫计算机化的病案系统或称基于计算机的病人记录(CPR, Computer-Based Patient Record)。它是用电子设备(计算机、健康卡等)保存、管理、传输和重现的数字化病人的医疗记录,取代了手写纸张病历。它的内容包括纸张病历的所有信息。

根据国家卫生部颁发的《电子病历基本架构与数据标准电子病历》,电子病历定义为:电子病历是医疗机构对门诊、住院患者(或保健对象)临床诊疗和指导干预的、数字化的医疗服务工作记录。

电子病历是用电子设备(计算机、健康卡等)保存、管理、传输和重现的数字化的病人医疗记录,取代手写纸张病历。电子病历具有主动性、完整和正确、知识关联、及时获取等特征,是医疗机构对门诊、住院患者(或保健对象)临床诊疗和指导干预的、数字化医疗服务工作记录。

21.5.2 应用安全需求

随着电子病历系统在医院的普遍使用,病历无纸化存储再也不是空谈了,消除纸张病历这一信息孤岛的思想已经深入人心。众所周知,病历不仅是病程记录,也是重要的、具

有法律效力的文件，在解决医患纠纷中有着不可替代的作用。正因如此，在全面无纸化建设的过程中，医院心存诸多顾虑，包括：电子病历中的电子签名是否获得法律认可、电子病历容易复制是否能保证法律依据的唯一性，担心电子形态的病历会成为患者投诉医院的依据。归根结底，就是如何全面解决电子病历与纸质病历具有同等的法律效力的问题。

当前医院中以病人为中心的 EMR 系统（电子病历系统）在应用过程中存在一些不安全因素：登录系统的身份合法性、电子病历的法律有效性、电子病历的正式性、电子病历信息的隐私保护、责任确定、时间取证的公正性、医院之间电子病历共享的安全等问题，这些不安全的因素阻碍着信息化的进一步发展。虽然部分医院的信息化系统建设已经走在前面，但依然保留着传统的病案管理方式，不仅没有解脱传统作业方式，还增加了医生的工作量，从而降低了诊疗效率。

若不解决上述问题，尤其是发生因电子病历引起的医疗纠纷时，无法站在第三方公正机构角度证明电子病历的真实合法性，进而无法起到公正性作用。

这些安全问题愈来愈被卫生管理部门所关注，随着卫生部出台卫生部电子认证相关法规（包括《卫生系统电子认证服务管理办法》、《卫生系统电子认证服务规范》等），以及电子病历相关规范（包括《病历书写基本规范》、《电子病历基本规范》等），营造安全可信的电子医疗卫生环境已成为大势所趋，在电子病历系统中采用数字证书技术进行安全认证能够保证其安全可信。

通过对医院业务特点、IT 现状及安全风险的分析，医院在安全认证方面的安全需求总结如下。

1. 医院信息系统的用户身份真实性需求

医院信息系统所处环境的封闭性导致其对身份认证强度的弱化。目前，医院医护人员登录信息系统普遍采用“用户名+口令”的方式，随着医院对信息系统的依赖程度逐步加深，这种弱认证方式的弊端逐渐凸显，它直接导致医生之间冒名顶替、实习医生代替主治医生出具诊断报告等情况，同时这种弱认证方式破解极为简单，很容易导致内部重要医疗数据的外泄，甚至导致医院信息系统遭受破坏性攻击。因此，建设具备高安全性、高可靠性的医院信息系统身份认证机制，保证登录医院信息系统用户身份的真实性，在目前医院信息系统建设过程中显得尤为迫切。

2. 医疗数据的责任归属问题

随着医院信息系统的各项功能逐步取代医院传统看病诊疗方式的同时，医护人员从对一张纸质诊断书的负责转向对一段数据电文描述内容的认可，数据电文的责任归属是否明确直接关系到信息化流程能否完全取代传统的纸质流程。因此，在用户身份真实可信的前提下，需要结合可靠的电子签名，建立医院信息系统中的责任认定机制，保障医疗数据明确的责任归属，从而消除医疗数据人工打印、手工签字的模式，实现真正意义的无纸化诊疗过程，使信息化的高效率优势充分发挥。

3. 医疗行为的时间可信需求

医疗管理中电子病历的生成、修改及访问等时间敏感性极高，然而，目前这些事件均由信息系统服务器时间产生，很容易发生时间记录不准确，从而导致医疗行为时间缺乏公信力，因此在保证时间源可信的前提下，需要对所有关键行为操作进行时间戳处理

并记录,确保提供可信的时间服务。

4. 数据在网络中完整传输问题

医院信息系统多数运行在医院内的局域网上,局域网上的医院信息系统终端与服务器的信息传输安全往往被忽视,因此,信息有可能被窃取并篡改,无法保障医务人员在系统终端上输入和浏览的电子病历信息的正确性。

21.5.3 应用安全总体架构

电子病历应用安全总体架构主要由3部分组成。

1. 卫生部数字证书服务管理系统

根据《卫生系统电子认证服务管理办法(试行)》的规定,卫生部将建设集中的数字证书服务管理系统,用于卫生系统内所有证书用户信息的收集、查询、统计和分析,以及进行用户意见收集、服务质量监督等管理工作。

卫生部通过数字证书服务管理系统对在卫生系统领域开展电子认证服务的CA机构实行接入控制及服务管理。拟为卫生系统领域提供服务的电子认证服务机构,须符合《卫生系统电子认证服务管理办法(试行)》的相关要求,将CA系统接入到卫生部数字证书服务管理系统。

2. CA中心

CA中心为医院电子病历提供电子认证服务,主要包括证书业务服务和技术支持服务。

证书业务服务主要包括证书管理、证书查询和时间戳服务等。证书管理主要包括证书申请、证书发放、证书更新、证书吊销、证书解锁、密钥恢复等业务服务。证书查询主要包括LDAP目录访问服务、OCSP证书在线状态查询服务及CRL证书黑名单列表下载服务。

技术支持服务主要包括使用帮助、应用咨询培训、应急保障和应用集成支持等。

CA中心有将数字证书申请、更新和吊销等相关信息同步到卫生部数字证书服务管理系统的功能。数据同步时,遵循和调用证书服务管理系统的证书信息同步接口。

3. 医院安全可信电子病历系统

医院安全可信电子病历系统主要由电子病历系统、安全认证系统、LRA与证书管理员、医护人员和科室机构证书等构成。

证书管理员通过LRA为医护人员、科室机构和内部设备发放数字证书。

安全认证系统主要包括密码模块、设备证书、时间戳、签名/验签、电子签章、签名存储等模块。

网上病历应用安全总体框架如图21-7所示。

21.5.4 网络部署结构

部署LRA系统,通过CA中心实现实时互联;为医院证书管理员颁发数字证书,介质为USBKey,进行各种证书业务操作,如证书申请、证书发放、证书更新、证书吊销等。

部署安全认证系统,配置证书注册模块、签名验签模块、签名存储模块、电子签章模块、时间戳模块;部署硬件密码设备,保证密钥存储和密码运算的安全性;在电子病历服务端系统部署并应用集成服务端安全接口模块,在电子病历客户端部署并应用集成客户端安全接口模块。部署数据库系统,用于存储电子签名相关数据。

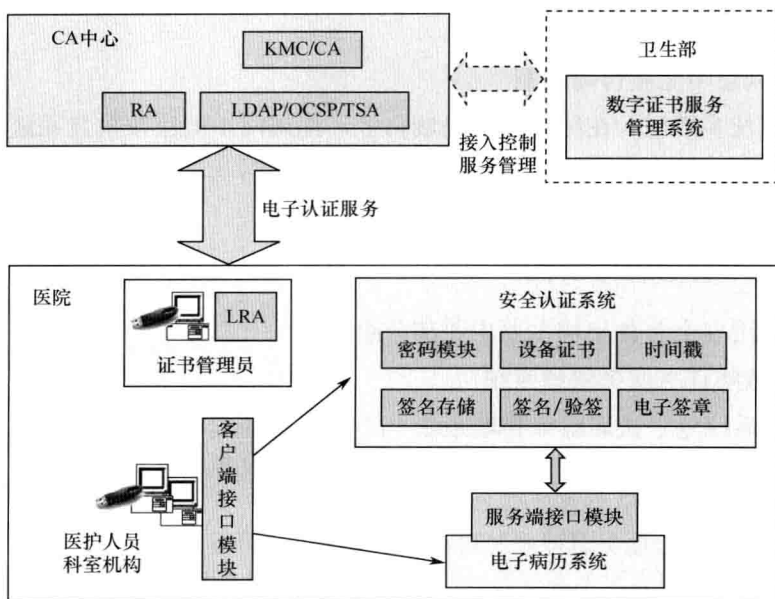


图 21-7 电子病历应用安全总体框架

为医护人员和科室机构签发数字证书，介质为 USBKey；为内部设备签发数字证书，介质为硬件密码设备。

具体部署如图 21-8 所示。

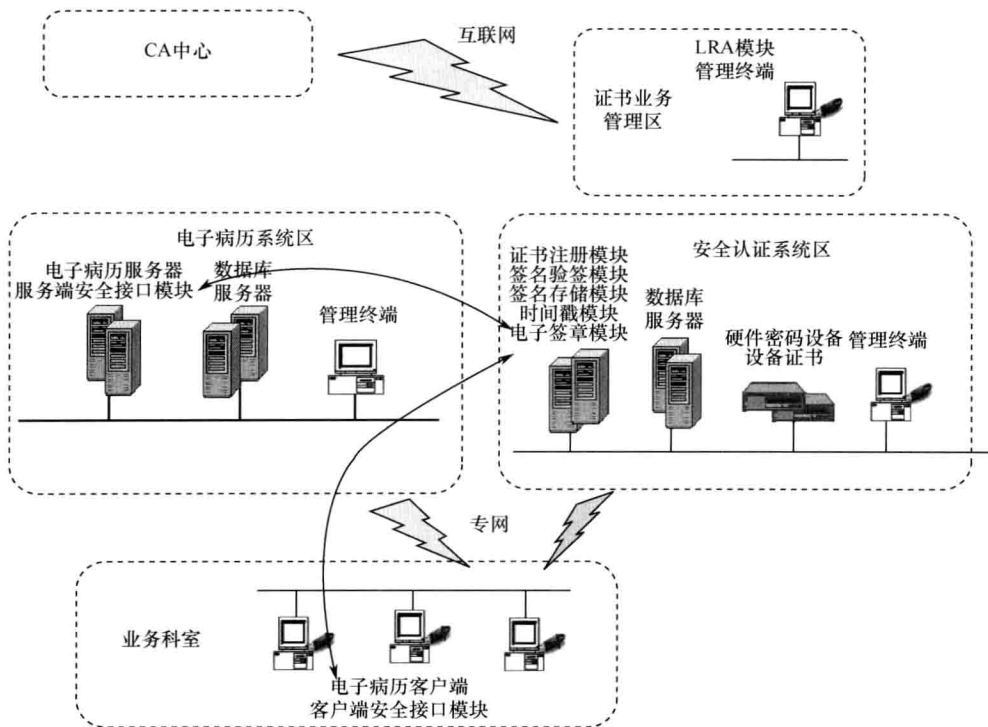


图 21-8 电子病历应用安全网络部署结构

21.6 公交 IC 卡在线充值系统

21.6.1 简介

2002 年, 建设部为了规范全国城市建设事业 IC 卡应用管理, 促进建设事业 IC 卡市场的健康、有序发展, 保障应用系统的安全性和兼容性组织, 制定并颁布了《建设事业 IC 卡应用技术》国家行业标准 (编号为 CJ/T166-2002), 从而使建设事业 IC 卡应用有章可循。

城市一卡通同样借鉴并发展了金融 IC 卡的成功技术和经验, 并根据公交应用的特殊需求, 进行了有针对性的扩展, 从而使公交卡更好地满足实际应用的需要, 更加方便地为广大市民服务。

城市一卡通在近几年得到了大规模应用。截至 2011 年, 已有 90 多个城市建立了不同规模和水平的公交 IC 卡应用系统, 累计发卡量 2000 余万张。经过几年的实践, 建设事业 IC 卡的应用形成了一些特点: 一是在公共交通领域实现了“一卡多用”; 二是双界面 CPU 卡已经在建设领域得到了具体应用; 三是实现了异地互通; 四是实现了跨行业的“一卡多用”。

与 PBOC 电子钱包/电子存折不同的是, 所有 IC 卡必须含有建设部申请的认证码, 才能使用建设事业领域各种应用系统。具体金融 IC 技术细节请参考《商业银行密码技术应用》“第四章 电子钱包/存折密码应用体系”。

21.6.2 应用安全需求

尽管公交 IC 卡使用非常广泛, 但由于公交卡属于匿名购买和无密码消费, 为减少公交卡丢失所造成的经济损失, 市民一般只在公交卡中存储少量资金, 消费完毕后再进行充值, 这就导致公交 IC 卡的频繁充值。但由于当前技术所限, 公交卡充值必须要到充值点进行人工充值, 给市民带来很多不便。由于受到成本限制, 公交卡充值点无法做到随处可见, 导致充值点人满为患。

因此通过互联网实现对公交 IC 卡的在线充值成为未来的发展趋势, 不仅可免去持卡人到充值点排队充值的麻烦, 同时也降低了公交卡服务机构建立充值点的成本。公交 IC 卡已采用对称密码体系, 只适合在相对封闭的环境下运行。

公交 IC 卡在线充值应用安全方面的具体需求主要包括:

- ① 正确鉴别 IC 卡的身份。保证 IC 身份的真实性和合法性。
- ② 保证交易数据的真实性和完整性。防止非法用户对数据进行假冒、篡改和删除, 防止数据传送过程中信息的丢失和重复, 保证信息传送次序的统一。
- ③ 保证交易数据的机密性。通过对一些敏感的数据进行加密来保护系统之间的数据交换, 防止除接收方之外的第三方截获数据。
- ④ 不修改 IC 卡现有的对称密钥体系。
- ⑤ 审计能力。根据机密性和完整性的要求, 对交易结果进行记录。

21.6.3 应用安全总体架构

公交 IC 卡在线充值应用安全总体框架如图 21-9 所示。

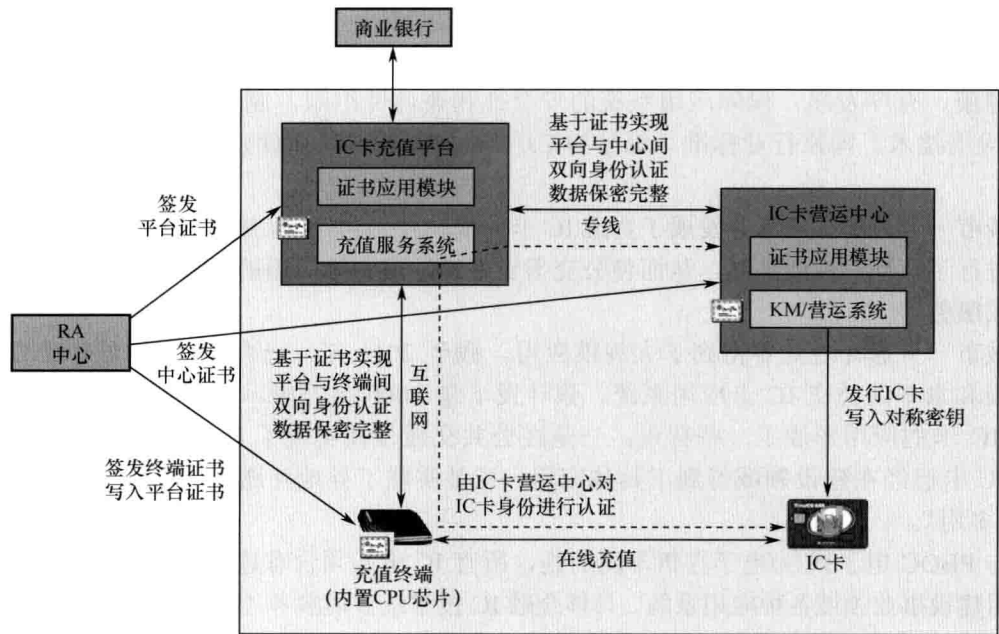


图 21-9 公交 IC 卡在线充值应用安全总体框架

IC 卡充值平台将商业银行和 IC 卡营运中心连接起来，通过互联网为充值终端提供公交卡充值服务。

为保证网络安全性，IC 卡营运中心只允许通过专线与 IC 卡充值平台进行连接。

为保证应用安全性，RA 中心为充值终端、IC 卡充值平台和 IC 卡营运中心签发数字证书，基于数字证书技术实现充值平台与充值终端、充值平台与营运中心之间的双向身份认证、数据保密性和完整性。

公交 IC 卡的所有交易都是基于对称密钥体系的，采用非常严格的密钥管理机制对对称密钥进行全过程管理。为保证对称密钥的安全性，充值终端和充值平台都不能对 IC 卡进行身份认证、充值交易等任何涉及对称密钥的操作，只允许 IC 卡营运中心对 IC 卡进行身份认证、充值交易等涉及对称密钥的操作。充值终端只是充当普通读卡器角色，充值平台只是作为网络传输通道，将 IC 卡与营运中心之间的交互数据进行上传下达而已。

21.6.4 充值交易流程

1. 传统充值交易流程

IC 卡传统充值交易流程包括 4 个步骤，如图 21-10 所示。其中，MAC1 由 IC 卡使用基于充值密钥产生过程的密钥生成，营运中心通过验证 MAC1 来确认 IC 卡的合法身份；MAC2 由营运中心使用相同的过程密钥产生，IC 通过验证 MAC2 来确认营运中心的合法

身份, 如果 MAC2 验证通过, 则 IC 自动更新卡片内金额信息; TAC 由 IC 卡使用 TAC 密钥生成, 在 IC 卡中保存一份, 同时传送给营运中心。

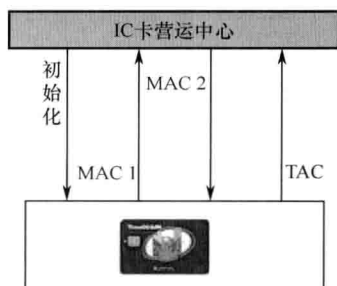


图 21-10 公交 IC 卡传统充值交易流程

2. 在线充值交易流程

为保证充值终端使用认可的数字证书, 需要对 IC 卡营运中心加密机进行以下升级改造:

- ① 支持 RSA 运算。
 - ② 内置 CA 证书。
 - ③ 新增在线充值指令: 完成验证终端签名、验证终端证书、验证 MAC1、产生 MAC2 并用终端证书加密。
- 并用终端证书加密。

在线充值交易流程如图 21-11 所示。

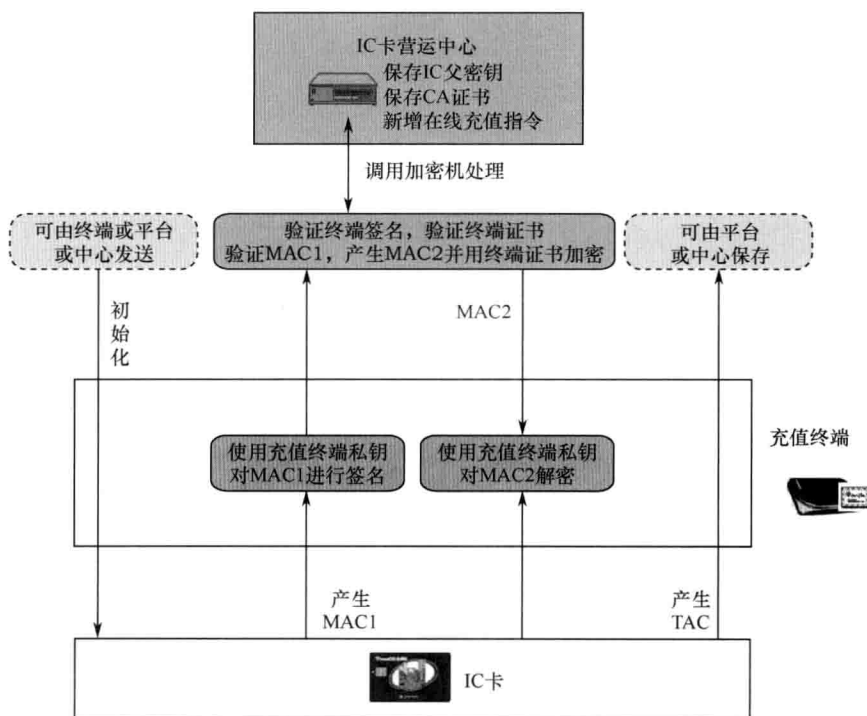


图 21-11 公交 IC 卡在线充值交易流程

- ① 向 IC 卡发出初始化充值指令：可由充值终端、充值平台或营运中心发起。
- ② IC 卡产生 MAC1 报文，回送给充值终端。
- ③ 充值终端使用终端私钥对 MAC1 报文进行签名，连同终端证书一起将签名后的 MAC1 报文发送给营运中心。
- ④ 营运中心系统调用加密机“在线充值指令”后，产生加密后的 MAC2 报文，发送给充值终端。
- ⑤ 充值终端接收到加密后的 MAC2 报文并使用私钥进行解密后，将 MAC2 报文发送给 IC 卡。
- ⑥ IC 卡接收到 MAC2 报文后，进行充值处理，然后产生 TAC 报文。
- ⑦ TAC 报文可由充值平台或营运中心保存。

第22章 实验四

22.1 Windows IIS 服务器证书配置

如果没有安装 IIS,需要先安装 IIS 组件。下面以 Windows 7 为例说明 IIS 组件的安装过程。

① 选择“控制面板”→“程序”→“程序和功能”→“打开或关闭 Windows 功能”，如图 22-1 所示。



图 22-1 安装 IIS 组件

② 选择“Internet 信息服务”，对需要的选项进行勾选。一个较小的选项集如图 22-2 所示。



图 22-2 IIS 安装选项

- ③ 单击“确定”按钮，等待安装完成。

22.1.1 下载并安装服务器证书

在 IIS 中启用 SSL 前，需要安装服务器证书，可以通过 IIS 证书管理功能完成证书申请和导入操作。

① 进入“控制面板”→“系统和安全”→“管理工具”→“Internet 信息服务 (IIS) 管理器”，出现 IIS 管理界面，如图 22-3 所示。



图 22-3 Internet 信息服务 (IIS) 管理器

- ② 在 IIS 管理界面单击“服务器证书”，进入服务器证书申请界面，如图 22-4 所示。

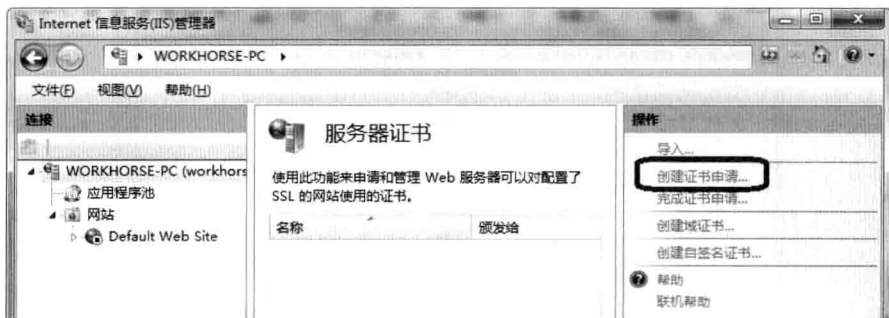


图 22-4 服务器证书管理

③ 单击“创建证书申请”功能链接，进入信息录入界面，如图 22-5 所示。在通用名称栏中输入 localhost（下文将以 localhost 为地址访问 IIS 服务器，应输入域名或 IP 地址），国家/地区中输入 CN（表示中国）。



图 22-5 证书使用者信息录入

④ 单击“下一步”按钮，出现“加密服务提供程序属性”界面，如图 22-6 所示，此处采用默认值。

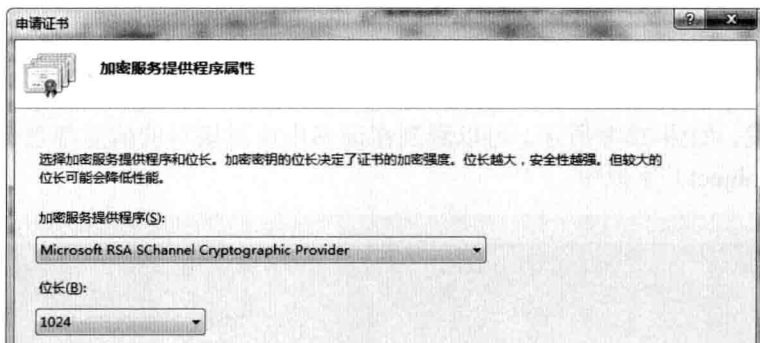


图 22-6 加密服务程序选项

⑤ 单击“下一步”按钮，出现证书请求文件保存路径输入界面，如图 22-7 所示，填入正确的文件路径。



图 22-7 证书请求文件保存路径

⑥ 单击“完成”按钮，生成的证书请求内容如下：

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIDUTCCARoCAQAwZjELMAkGA1UEBhMCQ04xEDAOBgNVBAGMB2JlaWppbmcxEDAO
BgNVBACMB2JlaWppbmcxDzANBgNVBAoMBnRlc3RjYTEOMAwwGA1UECwwFdGVzdDEx
EjAQBgNVBAMMCWxvY2FsaG9zdDCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEA
```

```

rl5wCYxSctD9FYLUStoeEse5byRSFaRvGZz/HMyhiKV/bghmBg/hNq+rWrvd/PF
Jzvy8ReocJfyoApj6ZuP5uIRpK172E0pav6wPhbWtvmDv85UfVW+XvixxSoS+oOI
CXFggNVxwlhgcP9FdWGc5v2KC3ThXAapphL1gd3ApdsCAwEAAaCCAakwGgYKKwYB
BAGCNw0CAzEMFgo2LjEuNzYwMS4yMEUGCSsGAQQBgjcVFDE4MDYCAQUMDHdvcmtob3JzZS1Q
QwwWd29ya2hvcnNILVBBDXhHdvcmtob3JzZQwLSW5ldE1nci5leGUwYK
KwYBBAGCNw0CAjFkMGICAQEeWgBNAGkAYwByAG8AcwBvAGYAdAAgAFIAUwBBACAA
UwBDAGgAYQBuAG4AZQBsACAAQwByAHkAcAB0AG8AZwByAGEAcABoAGkAYwAgAFAA
cgBvAHYAaQBkAGUAcgMBADCzWYJKoZIhvcNAQkOMYHBMIG+MA4GA1UdDwEB/wQE
AwIE8DATBgNVHSUEDDAKBggrBgEFBQcDATB4BgkqhkiG9w0BCQ8EazBpMA4GCCqG
SIb3DQMCAGIAgDAOBggqhkiG9w0DBAICAIAwCwYJYIZIAWUDBAEqMasGCWCGSAl
AwQBLTALBglghkgBZQMEAAQIwCwYJYIZIAWUDBAEFMAcGBSsOAwIHMAoGCCqGSIb3
DQMhMB0GA1UdDgQWBBRx8omRf7yXg6WV8NKriBB6WEQYZTANBgkqhkiG9w0BAQUF
AAOBgQAZ39LwmYNZ7Xjcw6HsVIw/GjvuMKvPFNVL7K1qCzv+Olj2NheZTp4+UBGu
gH0A642QwRqJZj0p5G9WgaHK4gtPZIGTEcI8ufGLcW5L/o2U1Dv9qopo/3Vp0yUa
KoTWv2u8WXcqMhTuROvC5Rbl2qJ1gCdjtlmVmHT9t78X+THi3w==
-----END NEW CERTIFICATE REQUEST-----

```

⑦ 产生了证书请求后，需要生成服务器证书，本例使用 OpenSSL CA 签发此服务器证书。执行命令“openssl.exe ca -in .\demoCA\iis_cert_req.txt -out iis_cert_req.crt -days 365”即可完成证书签发，如图 22-8 所示。可以看到在证书申请里填写的信息都包含在最后生成的证书使用者（subject）字段中。

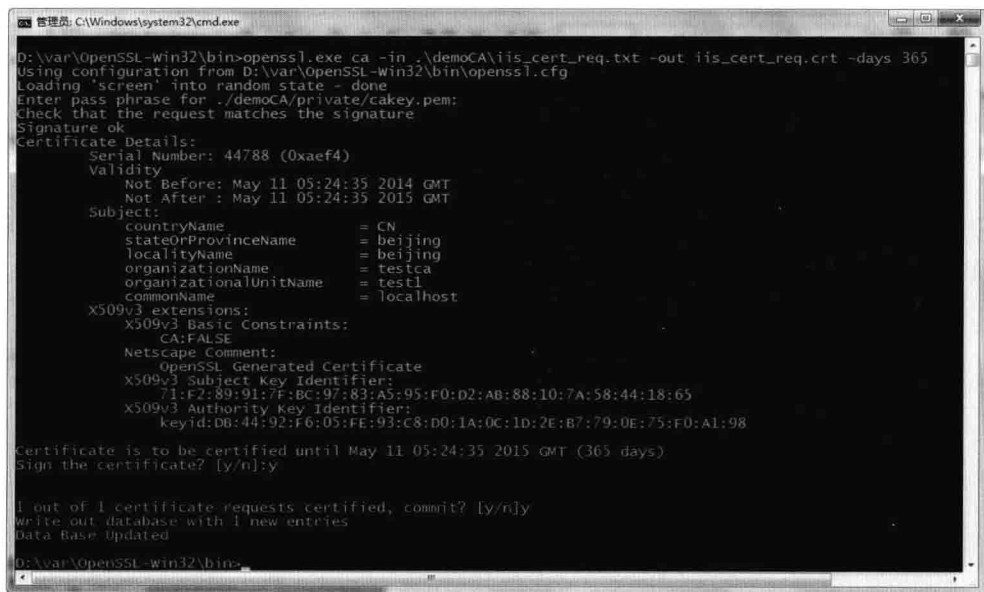


图 22-8 签发 IIS 服务器证书

⑧ 双击产生的服务器证书文件，显示信息如图 22-9 所示。

⑨ 证书签发完成后，需要导入到 IIS 服务器中，在图 22-4 中，单击“完成证书申请”功能链接，出现图 22-10 所示界面，输入证书存储路径和一个好记的名称，单击“确定”按钮后返回证书管理界面，如图 22-11 所示，可以看到导入的证书出现在列表中。



图 22-9 IIS 服务器证书信息

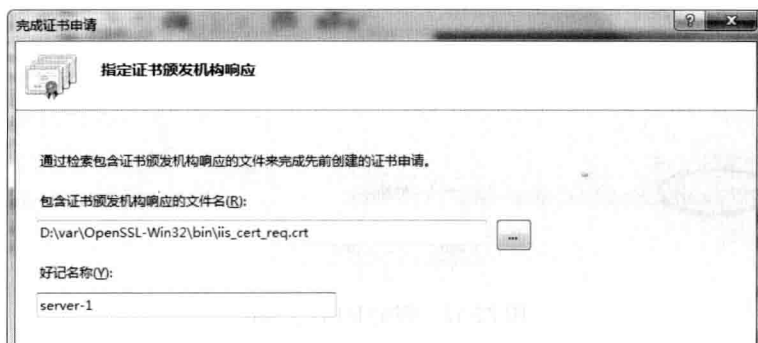


图 22-10 导入服务器证书



图 22-11 已导入服务器证书列表

⑩ 完成服务器证书导入后，需要配置 SSL 绑定端口。打开“Default Web Site”网站，

单击“绑定”链接，在出现的对话框中单击“添加”，出现添加网站绑定界面。依次选择“类型”为 https，“IP 地址”为“全部未分配”，“端口”为 443，“SSL 证书”为刚导入的 server-1，如图 22-12 所示。



图 22-12 绑定 HTTPS 端口

⑪ 单击“确定”按钮后，会出现警告信息，如图 22-13 所示。

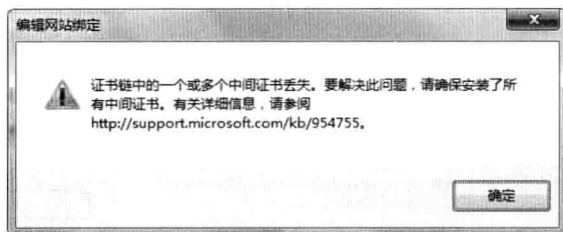


图 22-13 导入服务器证书警告信息

⑫ 该警告信息说明没有中间 CA 证书。按照提示，加入中间证书（即 CA 证书）。打开命令行窗口，输入 mmc.exe 命令，出现图 22-14 所示的操作界面。

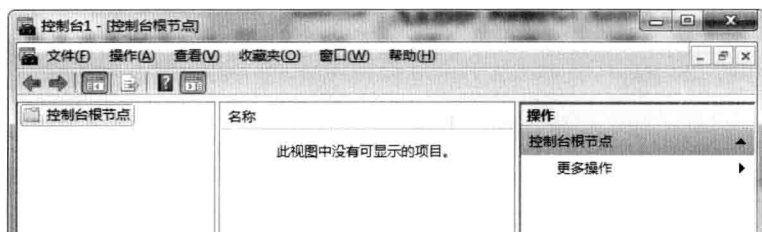


图 22-14 管理操作控制台

⑬ 在“文件”菜单中选择“添加/删除管理单元”，在出现的对话框中选择“证书”，然后单击“添加”按钮，如图 22-15 所示。

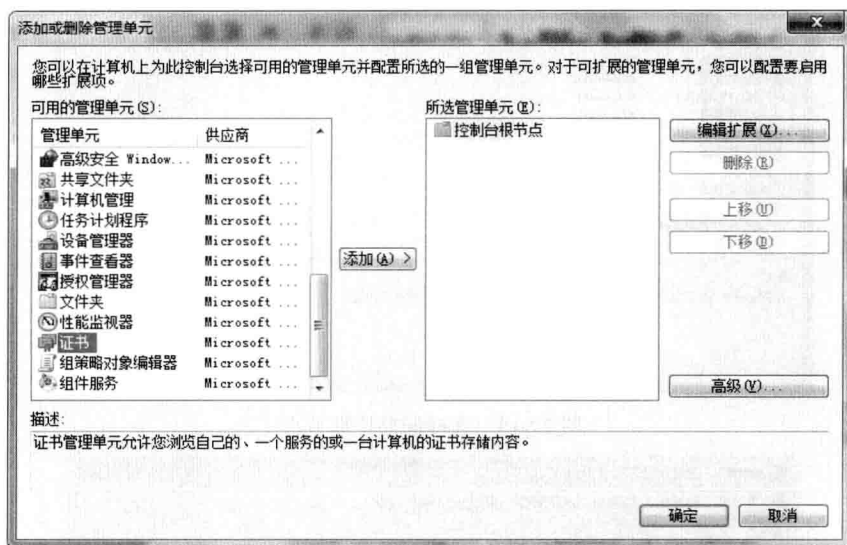


图 22-15 添加证书管理单元

⑭ 在出现的“证书管理”对话框中，选择“计算机帐户”，如图 22-16 所示，单击“下一步”按钮。

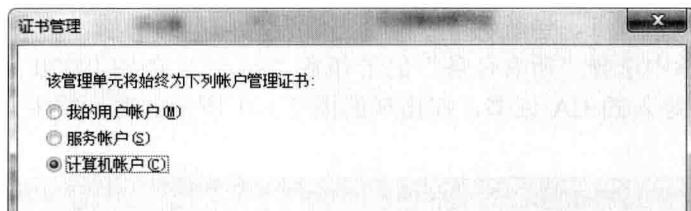


图 22-16 证书管理选择

⑮ 在出现的“选择计算机”对话框中，选择“本地计算机”，如图 22-17 所示，单击“完成”按钮。

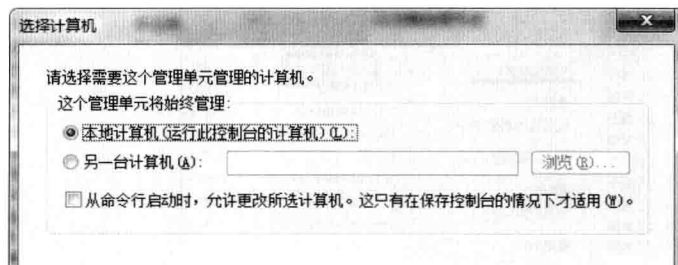


图 22-17 选择计算机

⑯ 出现图 22-18 所示界面，单击“确定”按钮，回到控制台界面，如图 22-19 所示。

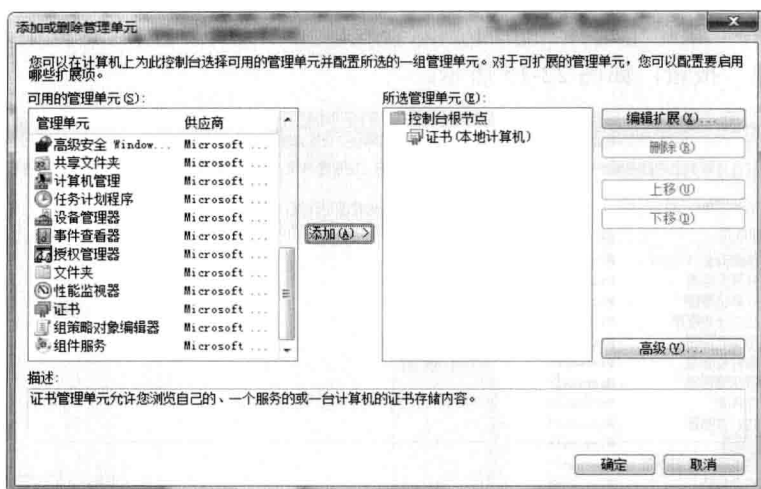


图 22-18 选择证书管理节点

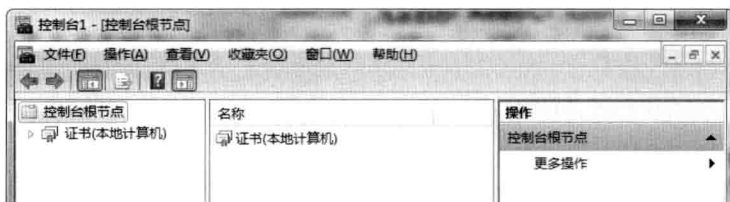


图 22-19 将证书管理节点加入到管理控制台

⑰ 展开“证书”节点，导航到“中级证书颁发机构”的子项“证书”，右键单击“证书”，在出现的菜单中选择“所有任务”的子任务“导入”，如图 22-20 所示。按照导航窗口的提示，选择要导入的 CA 证书，在出现的图 22-21 所示的对话框中，选择“中级证书颁发机构”。



图 22-20 选择导入证书

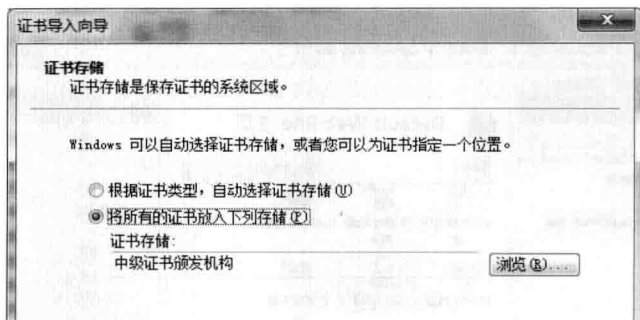


图 22-21 导入证书到“中级证书颁发机构”

⑱ 依次单击“下一步”按钮, 直到完成, 如图 22-22 所示。



图 22-22 导入 CA 证书到中级证书颁发机构

⑲ 至此完成证书安装。

22.1.2 配置 SSL 策略

① 在完成服务器端证书配置后, 需要配置 SSL 使用策略。打开“Internet 信息服务(IIS)管理器”, 在“Default Web Site”中选择“SSL 设置”, 如图 22-23 所示。

② 双击“SSL 设置”, 出现如图 22-24 所示的界面。勾选“要求 SSL”, 在“客户证书”中有 3 个选项, 分别是“忽略”、“接受”、“必需”:

- 忽略: 是默认选项。不要求客户端提供有效证书, 即使提供客户端证书, 也会忽略掉。该选项不会要求客户端在获得内容访问权限之前验证其身份。因此, 该设置在这些设置中的安全性最低。



图 22-23 SSL 配置选择



图 22-24 SSL 设置选项

- 接受：如果要接受客户端证书（若提供），并在允许客户端获得内容访问权限之前验证客户端身份，则选择该设置。
- 必需：要求客户端必须提供有效证书。选择该选项以在允许客户端获得内容访问权限之前要求证书验证客户端身份。

22.1.3 访问 Web Server

以上依次配置完成后，就可以访问支持 SSL 的 Web 了。重新启动 IIS 服务，在浏览器中输入访问地址 `http://localhost`，则显示图 22-25 所示的 IIS7 欢迎界面，表示配置 SSL 成功。



图 22-25 使用 https 访问 IIS 服务器

22.2 Apache 服务器证书配置

22.2.1 下载并安装服务器证书

在 Windows 环境下 Apache 的安装文件包括 32 位、64 位版本，以及 2.2.x 和 2.4.x 系列，可以从 <http://www.apachelounge.com/download/> 下载。下面以 2.4.x 为例说明服务器证书的配置，2.2.x 的服务器证书配置也类似。假定 Apache 的安装路径为 c:/Apache2。

除了 Windows 版本，Apache 使用最多的是在 Linux 环境。在 Linux 环境下的服务器证书配置与 Windows 下基本相同。

Apache 配置文件中使用的是文件证书和私钥，其 SSL 功能由 mod_ssl 模块提供，mod_ssl 模块利用 OpenSSL 实现了 SSL 功能。如果要使用密码设备提供的私钥和密码操作，需要配置专用密码设备的信息，因密码设备不同，其配置信息也会不同，本节不讨论专用密码设备配置信息，只说明使用文件方式配置 Apache 服务器证书。

Apache 安装完成后，形成如图 22-26 所示的目录结构。



图 22-26 Windows 下 Apache 安装后的目录

在 Apache 服务器中有一个总配置文件，即 `httpd.conf`，位于 Apache 安装目录下的 `conf` 子目录中，它配置了 Apache 的主要参数。为了便于管理，SSL 的配置单独放到了 `extra` 子目录下的 `httpd-ssl.conf` 中，要启用 SSL，需要在 `httpd.conf` 中找到如下行：

```
#LoadModule ssl_module modules/mod_ssl.so
#LoadModule socache_shmcb_module modules/mod_socache_shmcb.so
```

去掉 `LoadModule` 前面的“#”，在配置文件中以#开始的行表示是注释行，得到如下内容：

```
LoadModule ssl_module modules/mod_ssl.so
LoadModule socache_shmcb_module modules/mod_socache_shmcb.so
```

同时找到下面两行：

```
# Secure (SSL/TLS) connections
#Include conf/extra/httpd-ssl.conf
```

然后去掉 `Include` 前面的注释符“#”，得到如下内容：

```
# Secure (SSL/TLS) connections
Include conf/extra/httpd-ssl.conf
```

“`Include conf/extra/httpd-ssl.conf`”的意思是 `httpd-ssl.conf` 也会成为总配置文件的一部分，此处使用了文件相对路径，如果在前面加上 Apache 的安装路径就是文件的全路径（在 Apache 配置文件中，使用斜线（/）分隔文件路径，而不是使用反斜线（\））。然后打开 `httpd-ssl.conf` 文件，查看需要配置的内容。我们只关注证书相关内容，其他配置使用默认值（如使用 443 为 HTTPS 访问端口等）。

在 `httpd-ssl.conf` 中找到如下信息：

```
# Server Certificate:
# Point SSLCertificateFile at a PEM encoded certificate. If
# the certificate is encrypted, then you will be prompted for a
# pass phrase. Note that a kill -HUP will prompt again. Keep
# in mind that if you have both an RSA and a DSA certificate you
# can configure both in parallel (to also allow the use of DSA
# ciphers, etc.)
SSLCertificateFile "c:/Apache2/conf/server.crt"
```

此处要求配置服务器证书，证书格式为 PEM。继续往下找私钥配置信息：

```
# Server Private Key:
# If the key is not combined with the certificate, use this
# directive to point at the key file. Keep in mind that if
# you've both a RSA and a DSA private key you can configure
# both in parallel (to also allow the use of DSA ciphers, etc.)
SSLPrivateKeyFile "c:/Apache2/conf/server.key"
```

此处要求配置服务器的私钥，私钥可以用口令进行保护。

除了配置服务器证书和私钥外，还需要配置 CA 证书，找到如下信息：

```
# Certificate Authority (CA) :
# Set the CA certificate verification path where to find CA
# certificates for client authentication or alternatively one
# huge file containing all of them (file must be PEM encoded)
# Note: Inside SSLCACertificatePath you need hash symlinks
#       to point to the certificate files. Use the provided
#       Makefile to update the hash symlinks after changes.
SSLCACertificateFile " c:/Apache2/conf/ca.crt "
```

CA 证书同样以 PEM 格式表示，文件中可以包含多个 CA 证书，对于有多级证书的情况比较有用。

在使用 HTTPS 访问 Web 服务器时，要求 Web 网站的域名或 IP 地址必须是服务器证书的通用名 (commonName)。下面还是使用 OpenSSL CA 产生 Apache 服务器证书私钥，并配置完成 CA 证书。

1. 产生证书请求

执行命令产生证书请求：

```
openssl.exe req -newkey rsa:1024 -days 360 -keyout .\demoCA\private\server.key -keyform PEM
-out .\demoCA\server.req -outform PEM -nodes -subj "/C=CN/CN=localhost "
```

为了演示方便，产生的服务器私钥文件没有加密（在正式生产环境中，需要使用保护口令对私钥文件进行加密），私钥文件的内容为：

```
-----BEGIN PRIVATE KEY-----
MIICdQIBADANBgkqhkiG9w0BAQEFAASCAl8wggJbAgEAAoGBAKwY2ZD5dsOIAaa+
PxOYUrUPOT3JI5Gk3JzbH/0LuyH69VjsTd5moV+zyWg9qslIqa6IqwHLwP//ikly
WBNQ9BDz8tTH2Yq53m0C73ePF5NgQOExpTFZ8CPINQBjcojqOOn/DCnTwpVnC/E4
8psLTWTA0IJ0tt2WGaGUCYgn6GD7AgMBAAECgYBlune3fLVkDKYmAWBGt6i8O6LF
KauOeU2KPFBYcAy1X4kv+y0tP9ISz7feBbGXSw9aYwdhyunVRfj68Qenoh6CGcke
FE41EEDIRRLBJo6knhsiq99pVZSiWM/Fb6lsvbG/ceTwW4XPu9YKcRgrO6y2m/
1FrnhR6XhIZcWzSqqQJBAOBuFvdvee36v0lx1OZneA5I3TZGx1a2KalL1ke15gLC
ycwT4XXdd0TZcveujOY0xWOZZJev8MTBkdPY9sy+F58CQQDEZJp1qWymtqsu34BO
uLEqKIh2yKsQzUctRFj48hvmXdmoE1Nm4Q5unoGKug12iTDlrVNq51B/D1R9zlwZ
yaklAkBEAPIYUv/YZ6nxDCApFHatheMhYAVvwNsSSj4UEQ1ACwKXjfNMAq30PiL+
+HgYFSk9TzPSU/CeBLwLR3tRh9KrAkA9+imsfB0nt3nqPuo07aArV8NJCSbDFKUj
qfASEAWx+2gW3JJzYw605hyndPOOttjRgpNSp1EF6AaX9SmnkbZpAkB245moB9tc
EjDybU1gwX1PvnUKVnfVo8wtJT05eoHqH/bI+1e04sQUSjWxryPNLNVG0lawIYD
ZewJg+jS0mfJ
-----END PRIVATE KEY-----
```

2. 签发证书

执行命令签发证书：

```
openssl.exe ca -in .\demoCA\server.req -out server.crt -days 360
```

签发后的证书如图 22-27 所示。

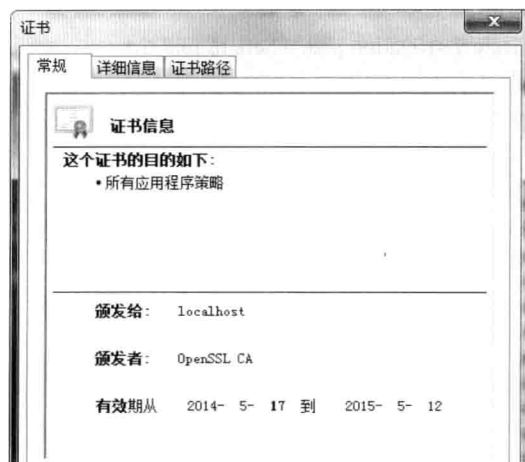


图 22-27 签发后的服务器证书

服务器证书的内容为:

-----BEGIN CERTIFICATE-----

```
MIICtzCCAZ+gAwIBAgIDAK71MA0GCSqGSIb3DQEBBQUAMCIXCzAJBgNVBAYTAKNO
MRMwEQYDVQQDDApPcGVuU1NMIENBMB4XDTE0MDUxNzAxNDgyOVVoXDTE1MDUxMjAx
NDgyOVVoITELMAkGA1UEBhMCQ04xEjAQBgNVBAMMCWxvY2FsaG9zdDCBnzANBgkq
hkiG9w0BAQEFAAOBJQAwYkCgYEAxBjZkPl2w4gBpr4/E5hStQ85PckjkaTcnNsf
/Qu7Ifr1WOxN3mahX7PJJaD2qyWWproirAcvA//+KSXJYE1D0EPPy1MfZirnebQLv
d48Xk2BA4TGIMVnwl8glAGNyiOo46f8MKdPCIWcL8TjymwtNZMDQgnS23ZYZoZQJ
iCfoYPsCAwEAAn7MHkwCQYDVR0TBAlwADAsBgIghkgBhvCAQ0EHxYdT3BlblNT
TCBHZW5lcmF0ZWQgQ2VydGlmaWNhdGUwHQYDVR0OBBYEFeh/ovg3V9omtDEz5qc3
mdLOOKTzMB8GA1UdIwQYMBaAFNtEkvYF/pPI0BoMHS63eQ518KGYMA0GCSqGSIb3
DQEBBQUAA4IBAQDF0rR3lyvT9PhRBHtrTkL+FTj0cgBtTyRe5kSs/cWwVWdu1MN6
F4bTmjwR22bjWs4P/QiRWDsEzK2neUXX5LdoJv6BwZsN39dduF40aGt5fk/lfeuJ
1rQDty90i09oiznAcd3GqXAPsvw6bQeElfoRTG5PgZYLcQOjN4aunumyMJIM0EI0
nWkt1g9yDd5HXxPd4FWQtYnfjAMAvzMQNR62ZMD5YWnaxc2Qv6sbSZaJgYvlBrO
Fz1T2l+19WiyGj6l9gZJRINBqBuVpNXJiUT4gbYIz29tpNzYDyRtMukSdtysuMv1
aZIP5oFdvH5qOerbmZdwETPaymFV33+gzwtb
```

-----END CERTIFICATE-----

CA 证书的内容为:

-----BEGIN CERTIFICATE-----

```
MIIDFzCCAF+gAwIBAgIJAMB4e3UzgUdVMA0GCSqGSIb3DQEBBQUAMCIXCzAJBgNV
BAYTAKNOMRMwEQYDVQQDDApPcGVuU1NMIENBMB4XDTE0MDUwODEzMzc1NFoXDTE0
MDUxNTEzMzc1NFowIjELMAkGA1UEBhMCQ04xEjAQBgNVBAMMCk9wZW5TU0wqQ0Ew
eggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDSeJ+BsTtVnH+o1534vBhq
Oe7nWpS9nkv13OGIXdqqdUIJQ2dOOnv40uM98wq/YDiLBhKgG7v0oVraZOUULI9u
```



```
pBlwWnz9FFY68WT7JxyxMdWkGb9/SFU83QkcY6HznFdeGbRLwk+7CcH1nwmfDwtO
cg0VeB2tkECUWLJ37C2MrxQrhwfgT6VuG78Wd5URY5PdshPSedAzSHEo74kdgkOx
hSSB4ghQB55YShgI5EanzHTGcnuB8yB58/IIISBug2PNFVMD9o+RKm3UfsoIgl37
8XX6AdwNmyEQae0TlvW+ZFqGQS05qauEq0sLX/CzaGRfN8OXExpLjpRTY5ZAzDzV
AgMBAAGjUDBOMB0GA1UdDgQWBbTbRjL2Bf6TyNAaDB0ut3kOdfChmDAfBgNVHSME
GDAWgBTbRjL2Bf6TyNAaDB0ut3kOdfChmDAMBgNVHRMEBTADAQH/MA0GCSqGSIb3
DQEBBQUAA4IBAQCbbRvZzYQWa1WFpIh3iQi76VoG1st1WQz71xNFxikdBiLCRPkS
+nhYnBXCv7UPesKYdeCDt4TKmYnfGQm4BdIl2glJIUENpndBWqB1IHQ9Xjr7/OfJ
+FLC6e3l1dmwrKRS3S4HxBWY/7vWBCHOXF2JzAZyqmgkWRG7UXb8ZppFiVXc94E8z
qgmEmQwv3AKqBzqlD/vosQswtYm7fyRn7Xj9sV4plNejKp6pBMIRr6KRnB98XYNG
N6o2nDw9kPS6KvFBkTgM8swmlh/impT0G7wU+GXe0U3toA9EarDCbhH0SjkVnPkf
BiBlexH3w5pBsO+i/PLK/ABQhQpjG+Y+FEgp
-----END CERTIFICATE-----
```

把服务器证书、服务器私钥、CA 证书文件复制到 Apache 的配置目录 C:/Apache2/conf/下。

22.2.2 配置 SSL 策略

对 SSL 策略的配置，同样在 httpd-ssl.conf 中实现，下面分别予以说明。

1. 支持的 SSL 协议

使用 SSLProtocol 配置，可以配置支持 SSLv2、SSLv3、TLSv1、all，如果要去掉对某一协议的支持，则在协议名称前添加减号“-”，如：

```
# 启用 SSLv3 和 TLSv1 但禁用 SSLv2
```

```
SSLProtocol all -SSLv2
```

all 表示全部 SSL 协议。因为 SSLv2 协议基本上不使用了，因此可以在配置文件中禁用它。

2. 使用算法

通过 SSLCipherSuite 指定在 SSL 握手时支持的算法。配置方式为：

```
SSLCipherSuite HIGH:MEDIUM:!aNULL:!MD5
```

SSLCipherSuite 指令的值是一个用冒号分隔的 OpenSSL 加密算法集字符串，用于在 SSL 握手过程中进行加密算法协商时告诉客户端允许使用哪些加密算法。

OpenSSL 加密算法集实际上是由 4 个属性组成的：

- ① 密钥交换算法：RSA 或 Diffie-Hellman 算法的各种变种。
- ② 认证算法：RSA，Diffie-Hellman，DSS 或 none。
- ③ 加密算法：AES，DES，Triple-DES，RC4，RC2，IDEA 或 none。
- ④ 摘要算法：MD5，SHA 或 SHA1。

可以指定加密算法集中每个属性的算法，也可以使用别名指定一组特定的算法集，如表 22-1 所示。

表 22-1 算法类型及选项

类型	选项	类型	选项
密钥交换算法	kRSA: 纯 RSA 密钥交换 kDhR: 使用 RSA 密钥的 Diffie-Hellman 密钥交换 kDhD: 使用 DSA 密钥的 Diffie-Hellman 密钥交换 kEDH: 临时 Diffie-Hellman 交互密钥	摘要算法	MD5: MD5 摘要 SHA1: SHA1 摘要 SHA: SHA 摘要
认证算法	aNULL: 不进行认证 aRSA: RSA 认证 aDSS: DSS 认证 aDH: Diffie-Hellman 认证	别名	SSLv2: 所有 SSLv2 算法 SSLv3: 所有 SSLv3 算法 TLSv1: 所有 TLSv1 算法 EXP: 所有出口算法 EXPORT40: 所有 40-位出口算法 EXPORT56: 所有 56-位出口算法 LOW: 所有低强度算法（非出口算法，DES） MEDIUM: 所有 128-位加密算法 HIGH: 所有使用 Triple-DES 或更高强度的算法 RSA: 所有使用 RSA 密钥交换的算法 DH: 所有使用 Diffie-Hellman 密钥交换的算法 EDH: 所有使用临时 Diffie-Hellman 密钥交换的算法 ADH: 所有使用匿名 Diffie-Hellman 密钥交换的算法 DSS: 所有使用 DSS 认证的算法 NULL: 所有不加密的算法
加密算法	eNULL: 不加密 AES: AES 加密 DES: DES 加密 3DES: Triple-DES 加密 RC4: RC4 加密 RC2: RC2 加密 IDEA: IDEA 加密		

- 可以使用下面的语法增删算法以及确定在握手阶段协商的“算法集”优先级顺序。
- ① [没有标记]: 向列表中增加一个算法集。
 - ② +: 在列表中的相应位置增加一个算法集。
 - ③ -: 从列表中临时删除相应的算法集（之后还可以被再次添加）。
 - ④ !: 从列表中永久删除相应的算法集（之后不可以被再次添加）。

该指令默认值为“ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP”，其含义为：首先永久删除所有使用匿名 Diffie-Hellman 密钥交换的算法，然后添加使用 RC4 和 RSA 的算法，再后顺序添加高、中、低强度的算法，最后再追加所有的 SSLv2 算法和出口算法到列表结尾。

可以使用“openssl ciphers -v”命令查看所有可用的“加密算法集”。

3. 客户端验证

使用 SSLVerifyClient 配置客户端验证策略，可以取值为 none、optional、require。none 表示不需要客户端证书，optional 表示客户端可以提供有效证书，require 表示客户端必须提供有效证书。

对于单向 SSL，可以配置为 none 或 optional，对双向 SSL 必须配置为 require。如“SSLVerifyClient require”表示要验证客户端证书。

在对客户端证书进行验证的情况下，还需要配置验证级别。在多级 CA 证书情况下，必须进行配置，否则会导致 SSL 握手失败。配置指令用 SSLVerifyDepth。简单说，SSLVerifyDepth 指定了验证证书链的长度。例如，“SSLVerifyDepth 10”表示配置最大验证证书链长度为 10。

22.2.3 访问 Web Server

完成配置后，重新启动 Apache 服务器进程，打开浏览器访问 Web 服务器，显示如图 22-28 所示的内容。



图 22-28 使用 SSL 访问 Apache 网站

从图 22-28 可以看到，客户端与服务器端完成了 SSL 握手，使用了 128 位对称加密算法，且使用了 TLS1.0 协议，CA 证书为 OpenSSL CA。

22.3 Tomcat 服务器证书配置

22.3.1 下载并安装服务器证书

从 <http://tomcat.apache.org/> 下载 7.0 版本的 Tomcat 并进行安装。假定 Tomcat 安装在 D:\tomcat7 目录下。配置 Tomcat 的安全服务包括两部分，一是产生服务器证书，二是配置 SSL 连接器。

1. 产生服务器证书

(1) 第一步，生成服务器私钥，产生密钥库文件

使用 Java 提供的 keytool 工具，在安装目录的 conf 子目录下执行命令：

```
keytool -genkey -alias tcserver -keypass test123 -keyalg RSA -keysize 1024 -validity 365 -keystore tcssl.keystore -storepass pass123 -dname "CN=127.0.0.1, C=CN"
```

参数含义请参考“12.3 JCA/JCE”的“使用证书”章节。在当前目录下产生了 tcssl.keystore 密钥库文件，包含服务器的私钥。

(2) 第二步，生成证书请求文件

执行命令：

```
keytool -certreq -alias tcserver -sigalg SHA1withRSA -file servercert.req -keystore tcssl.keystore -storepass pass123 -keypass test123
```

在当前目录下产生了 servercert.req 文件，使用 OpenSSL CA 对此证书请求进行签发。

(3) 第三步，签发服务器证书

执行命令：

```
D:\var\OpenSSL-Win32\bin\openssl.exe ca -in .\servercert.req -out servercert.crt -days 360
```

产生的证书如图 22-29 所示。

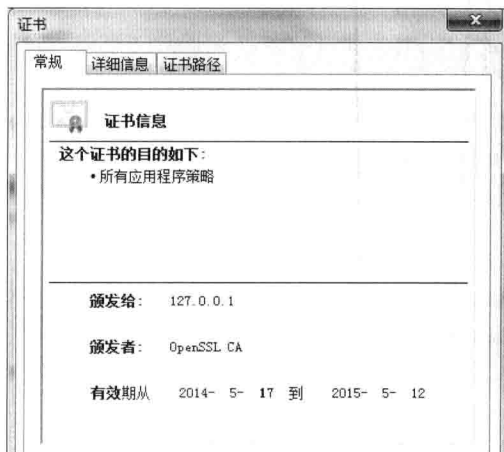


图 22-29 服务器证书信息

(4) 第四步，先导入 CA 证书，然后导入签发完成的服务器证书。

执行如下命令，可导入 CA 证书：

```
keytool -import -alias caroot -trustcacerts -keystore tcssl.keystore -storepass pass123 -file D:\var\OpenSSL-Win32\bin\demoCA\cacert.pem
```

结果如图 22-30 所示，表示导入证书成功，其中当提示是否信任此证书时，选择 y。

使用如下命令，可导入服务器证书：

```
keytool -import -alias tcserver -keystore tcssl.keystore -storepass pass123 -keypass test123 -file servercert.crt
```

结果如图 22-31 所示，表示证书导入到密钥库中。

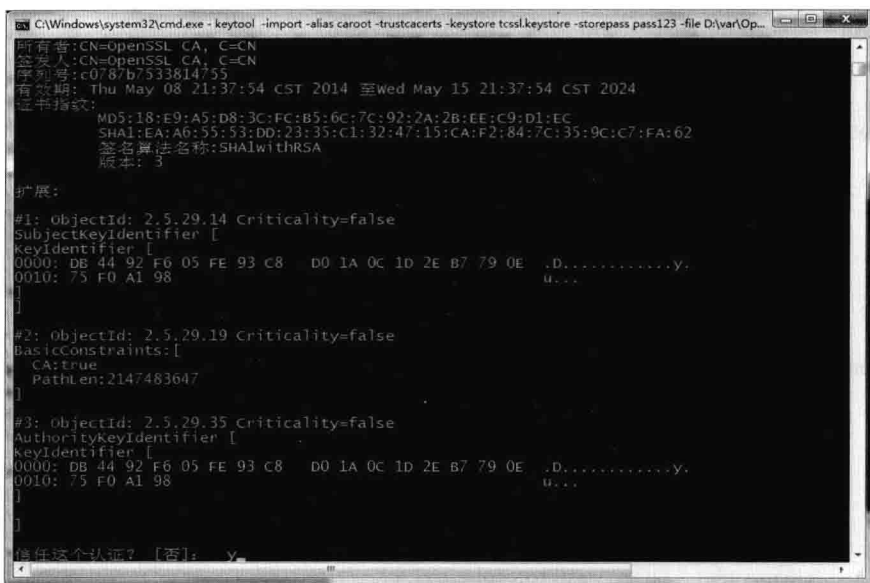


图 22-30 导入 CA 证书

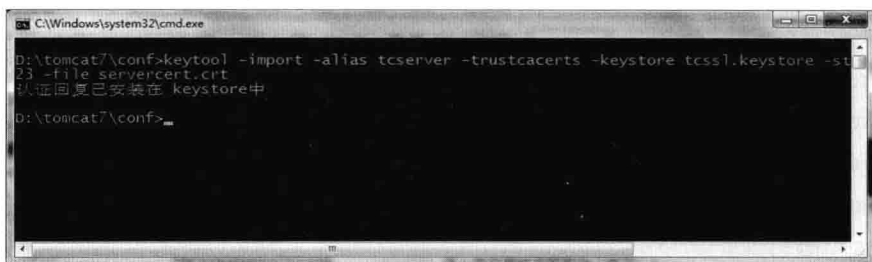


图 22-31 将服务器证书导入到密钥库中

导入服务器证书时，服务器证书的别名必须和私钥别名一致。请留意导入 CA 证书和导入服务器证书时的提示信息，如果在导入服务器证书时使用的别名与私钥别名不一致，系统将提示“认证已添加至 keystore 中”而不是应有的“认证回复已安装在 keystore 中”。

证书导入完成后再次查看 keystore 文件内容，执行命令：

```
keytool -list -keystore tcssl.keystore -storepass pass123
```

结果如图 22-32 所示，表示 tcserver 有对应的证书和私钥。

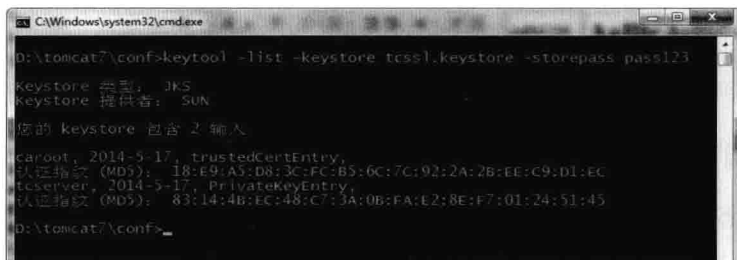


图 22-32 密钥库中存在证书

2. 配置 SSL 连接器

打开 conf 目录下的 server.xml 文件，找到并修改以下内容：

```
<!--
<Connector port=" 8443 " protocol=" org.apache.coyote.http11.Http11Protocol "
    maxThreads=" 150 " SSLEnabled=" true " scheme=" https " secure=" true "
    clientAuth=" false " sslProtocol=" TLS " />

-->
```

修改为

```
<Connector port=" 8443 " protocol=" org.apache.coyote.http11.Http11Protocol "
    maxThreads=" 150 " SSLEnabled=" true " scheme=" https " secure=" true "
    keystoreFile=" conf\tcssl.keystore " keystorePass=" pass123 "
    keystoreType=" JKS "
    keyAlias=" tcserver " keyPass=" test123 "
    clientAuth=" false " sslProtocol=" TLS " />
```

可见增加了密钥库文件位置、访问口令、类型，并指定了服务器密钥别名、访问口令参数。其中：

keystoreFile：指定密钥库的位置；

keystorePass：指定密钥库的口令；

keystoreType：指定密钥库的类型，支持 JKS、PKCS#12；

keyAlias：使用的密钥别名；

keyPass：密钥保护口令。

22.3.2 配置 SSL 策略

在 Tomcat 的 SSL 配置中，有 3 个选项与 SSL 握手有关，分别是 sslProtocol、ciphers、clientAuth。

① sslProtocol 指定使用的握手协议，在前面的配置中使用了 TLS 协议，也可以使用 SSLv2、SSLv3、TLSv1、TLSv1.1、TLSv1.2 协议。

② ciphers 指定允许使用的密码算法，本参数使用 JSSE 密码算法名称，多个算法之间以逗号分隔。

③ clientAuth 指定是否验证客户端证书。true 表示客户端必须提供有效证书；want 表示最好提供客户端证书，如果不提供也不算错；false 表示不要求客户端提供证书。

22.3.3 访问 Web Server

在完成上述配置后，重新启动 Tomcat，然后访问 https://127.0.0.1:8443，结果如图 22-33 所示界面，表示 SSL 配置成功，在图中显示了连接的信息。

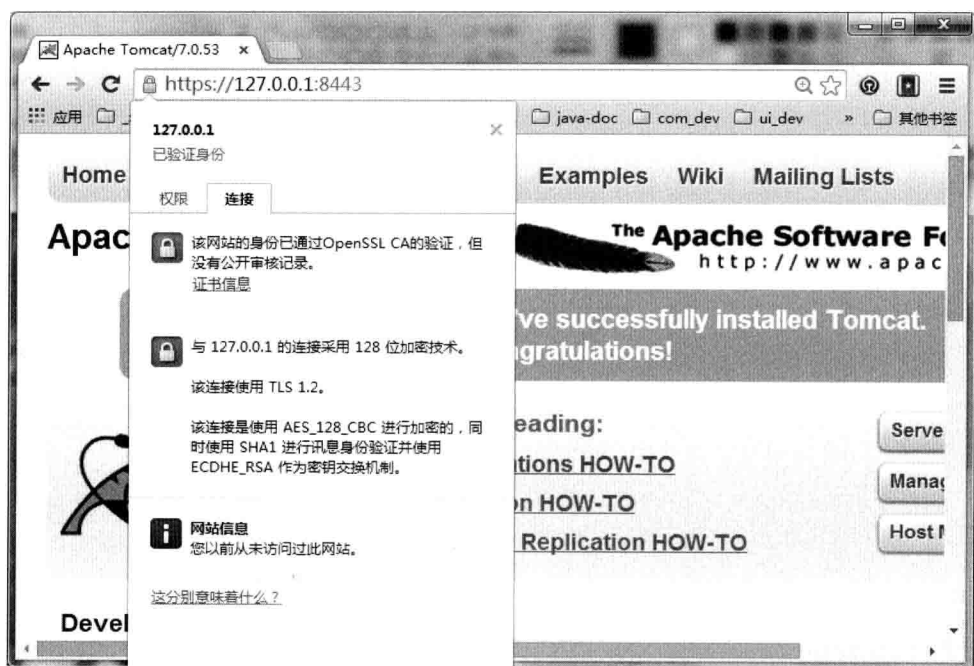


图 22-33 SSL 访问成功界面

第六部分

PKI 之运营：CA 中心

第23章 机房建设

《证书认证系统密码及其相关安全技术规范》规定了证书认证中心和密钥管理中心建设的基本要求。

23.1 业务系统

业务系统主要包括 CA 系统和 KMC 系统。CA 系统部署在证书认证中心，KMC 系统部署在密钥管理中心。

23.1.1 证书认证中心

1. 功能要求

CA 系统提供的服务功能主要有：

- ① 提供各种证书在其生命周期中的管理服务。
- ② 提供 RA 的多种建设方式，RA 可以全部托管在 CA 系统；也可以部分托管在 CA，部分建在远端。
- ③ 提供人工审核或自动审核两种审核模式。
- ④ 支持多级 CA 认证。
- ⑤ 提供证书查询、证书状态查询、证书作废列表下载、目录服务等功能。

2. 性能要求

CA 系统的性能应满足如下要求：

- ① 系统对用户接口采用标准的 HTTP、HTTPS 和 LDAP 协议，确保各种用户都能够使用本系统服务。
- ② 系统各模块的状态信息保存在配置文件和数据库内部，保证系统的部署方便性和配置方便性，当系统需改变配置时无须中断系统的服务。
- ③ 各模块的功能可以通过配置文件进行控制，系统可以根据不同的需求进行设置。
- ④ 系统某一功能模块可有多个实例，并且多个实例可运行在一台或多台计算机上。
- ⑤ 系统应有冗余设计，保证系统的不间断运行。

3. 管理员配置要求

CA 应设置下列管理和操作人员：超级管理员、审计管理员、业务管理员、业务操作员。其中，“超级管理员”负责 CA 系统的策略设置，设置各子系统的业务管理员并对其管理的业务范围进行授权。“业务管理员”负责 CA 系统的某个子系统的业务管理，设置本子系统的业务操作员并对其操作的权限进行授权。“业务操作员”按其权限进行具体的业务操作。“审计管理员”负责对涉及系统安全的事件和各类管理和操作人员的行为进行

审计和监督。

上述各类人员使用证书进行登录，其中“超级管理员”和“审计管理员”的证书应在CA系统进行初始化时同时产生。

另外，CA应设置安全管理员，全面负责系统的安全工作。

4. 网络划分

CA系统的计算机网络需要合理分段，原则上要求整个网络划分为4部分。

① 公共部分（区）：CA用户所在的网络，所有用户将通过该网络访问CA。

② 服务部分（区）：为外部用户提供域名解析功能，并负责内部系统对外邮件的收发功能。包括系统的各种Web服务器和从目录服务器，是外部用户访问内部功能的接口，为用户提供访问界面。

③ 管理部分（区）：仅供CA的工作人员使用的网络。

④ 核心部分（区）：包括各种核心应用、数据库和密码设备等在内的实现系统功能的安全网络。

当RA采用客户机/服务器（C/S）模式时，应该按照上述方式划分网络，如图23-1所示。

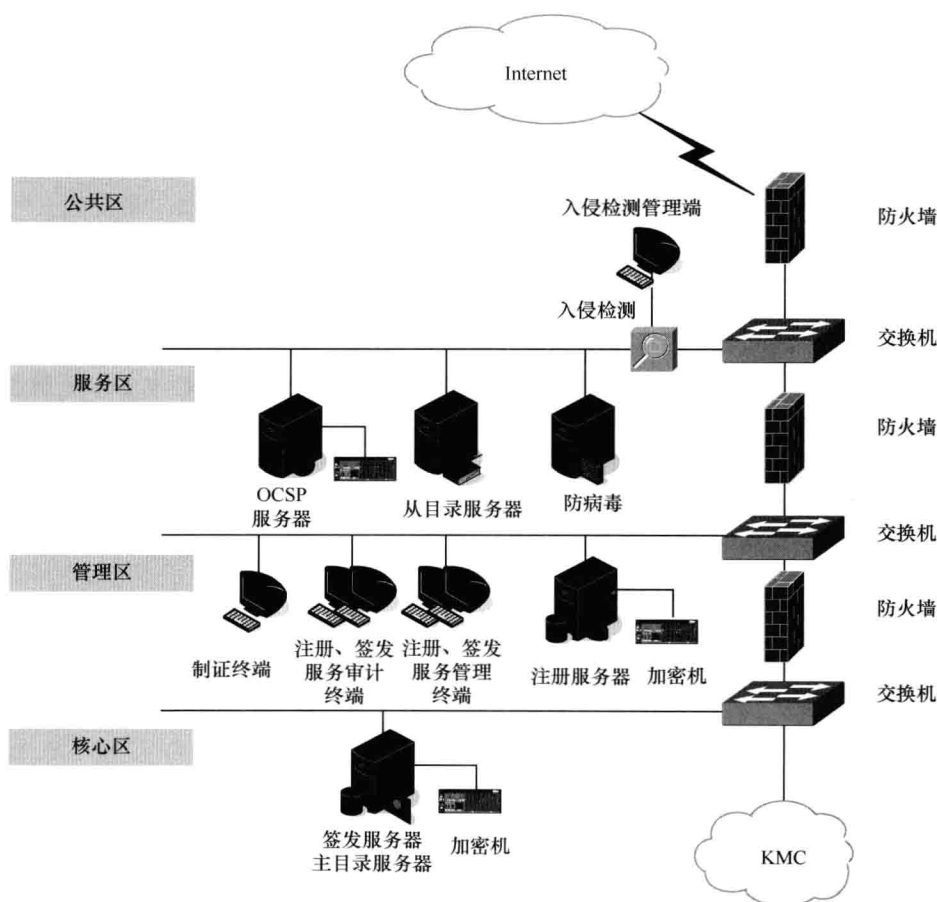


图 23-1 RA 采用 C/S 模式时 CA 的网络结构示意图

当 RA 采取浏览器/服务器 (B/S) 模式时, 可将服务区与管理区网络放在同一网段, 如图 23-2 所示。

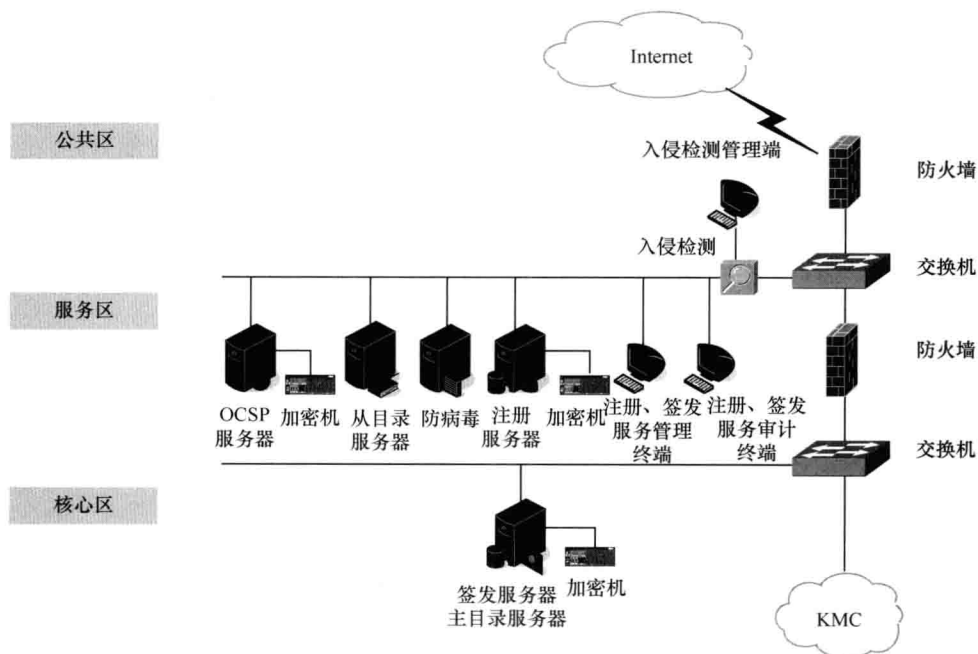


图 23-2 RA 采用 B/S 模式时 CA 的网络结构示意图

5. 初始化要求

CA 的初始化过程必须完成下列工作：

- ① 产生本 CA 的机构密钥并进行安全备份。
- ② 若本 CA 为根 CA, 则使用根 CA 的签名密钥进行自签名; 若本 CA 从属于某一根 CA, 则将产生的签名公钥提交根 CA 签发本 CA 的证书。
- ③ 由 CA 签发 CA 服务器证书。
- ④ 由 CA 签发 RA 服务器证书 (可选)。
- ⑤ 由 CA 签发超级管理员和审计管理员证书。
- ⑥ 由 CA 签发其他管理员和操作员证书。

23.1.2 密钥管理中心

密钥管理中心的工程建设按照与 CA 统一规划、有机结合、独立设置、分别管理的原则进行。

1. 功能要求

密钥管理中心应提供下列服务功能：

- ① 为 CA 提供密钥生成服务；
- ② 为司法机关提供密钥恢复服务；

③ 为用户提供密钥更新、密钥恢复、密钥作废服务。

2. 性能要求

密钥管理中心性能应满足如下要求：

① 密钥的保存期应大于 10 年。

② 系统应支持多并发服务请求。

③ 系统各模块的状态信息保存在配置文件和数据库内部，保证系统的部署方便性和配置方便性，当系统需改变配置时无须中断系统的服务。

④ 各模块的功能可以通过配置文件进行控制，系统可以根据不同的需求进行设置。

⑤ 系统应有冗余设计，保证系统的不间断运行。

3. 管理员配置要求

KMC 应设置下列管理和操作人员：超级管理员、审计管理员、业务管理员、业务操作员。其中，“超级管理员”负责 KMC 系统的策略设置，设置各子系统的业务管理员并对其管理的业务范围进行授权。“业务管理员”负责 KMC 系统的某个子系统的业务管理，设置本子系统的业务操作员并对其操作的权限进行授权。“业务操作员”按其权限进行具体的业务操作。“审计管理员”负责对涉及系统安全的事件和各类管理与操作人员的行为进行审计和监督。

上述各类人员使用证书进行登录，其中“超级管理员”和“审计管理员”的证书应在 KMC 系统进行初始化时同时产生。

另外，KMC 应设置安全管理员，全面负责系统的安全工作。

4. 初始化要求

KMC 的初始化过程必须完成下列工作：

① 生成 KMC 的机构密钥并进行安全备份。

② 由授权的 CA 签发 KMC 服务器证书。

③ 由授权的 CA 签发超级管理员和审计管理员证书。

④ 由授权的 CA 签发业务管理员和业务操作员证书。

23.2 应用安全

CA 与 KMC 系统的应用安全包括系统安全、通信安全、密钥安全、证书管理安全、安全审计等各方面的安全。

1. 系统安全

系统安全的主要目标是保障网络、主机系统、应用系统及数据库运行的安全，应采取防火墙、病毒防治、漏洞扫描、入侵监测、数据备份、灾难恢复等安全防护措施。

2. 通信安全

通信安全的主要目标是保障 CA 系统各子系统之间、CA 与 KMC 之间、CA 与 RA 之间的安全通信，应采取通信加密、安全通信协议等安全措施。

3. 密钥安全

密钥安全的主要目标是保障 CA 系统中所使用的密钥，在其生成、存储、使用、更新、废除、归档、销毁、备份和恢复等整个生命周期中的安全，应采取硬件密码设备、密钥管理安全协议、密钥存取访问控制、密钥管理操作审计等多种安全措施。

(1) 基本要求

密钥安全的基本要求是：

- ① 密钥的生成和使用必须在硬件密码设备中完成。
- ② 密钥的生成和使用必须有安全可靠的管理机制。
- ③ 存在于硬件密码设备之外的所有密钥必须加密。
- ④ 密钥必须有安全可靠的备份恢复机制。
- ⑤ 对密码设备操作必须由多个操作员实施。

(2) 根 CA 密钥

根 CA 密钥的安全性除了满足基本要求外，还应满足下列要求：

① 根 CA 密钥的产生。

CA 系统的根密钥由硬件密码设备生成并存放在该密码设备中，应采用密钥分割或秘密共享机制进行备份。保存分割后的根密钥的人员称为分管者。

生成根 CA 密钥时，应先选定分管者，数量可以限定为 3 个或 5 个。选定的分管者应分别用自己输入的口令保护分管的密钥，分管的密钥应存放在智能 IC 卡或智能密码钥匙中。智能 IC 卡或智能密码钥匙也应备份，并安全存放。

根 CA 密钥的产生过程必须进行记录。

② 根 CA 密钥的恢复。

恢复根 CA 密钥时，要有满足根 CA 密钥恢复所必需的分管者人数。各个分管者输入各自的口令和分管的密钥成分在密码设备中恢复。

③ 根 CA 密钥的更新。

更新根 CA 密钥时，需重新生成根 CA 密钥，其过程同根 CA 密钥的产生。

④ 根 CA 密钥的废除。

根 CA 密钥的废除应与根 CA 密钥的更新同步。

⑤ 根 CA 密钥的销毁。

根 CA 密钥应与备份的根 CA 密钥一同销毁。由密码主管部门授权的机构实施。

(3) 非根 CA 密钥

非根 CA 密钥的安全性要求与根 CA 密钥的安全性要求一致。

(4) 管理员证书密钥

管理员包括超级管理员、审计管理员、业务管理员和业务操作员等。管理员证书密钥应由证书载体来产生和存储。

管理员证书密钥的安全性应满足下列要求：

- ① 管理员证书密钥的产生和使用必须在证书载体中完成。
- ② 密钥的生成和使用必须有安全可靠的管理机制。
- ③ 管理员的口令长度为 8 个字符以上。

- ④ 管理员的账号要和普通用户账号严格分类管理。

4. 证书管理安全

证书的管理安全应满足下列要求:

- ① 验证证书申请者的身份。
- ② 防止非法签发和越权签发证书, 通过审批的证书申请必须提交给 CA, 由 CA 签发与申请者身份相符的证书。
- ③ 保证证书管理的可审计性, 对于证书的任何处理都需要做日志记录。通过对日志文件的分析, 可以对证书事件进行审计和跟踪。

5. 安全审计

CA 系统在运行过程中涉及大量功能模块之间的相互调用, 以及各种管理员的操作, 对这些调用和操作需要以日志的形式进行记载, 以便用于系统错误分析、风险分析和安全审计等工作。

(1) 功能模块调用日志

系统内的各功能模块在运行过程中会调用其他功能模块或被其他功能模块所调用, 对于这些相互之间的功能调用, 各模块应该记录如下数据:

- ① 调用请求的接收时间。
- ② 调用请求来自的网络地址。
- ③ 调用请求发起者的身份。
- ④ 调用请求的内容。
- ⑤ 调用请求的处理过程。
- ⑥ 处理结果等。

(2) CA 系统管理员审计

CA 系统管理员的下列操作应被记录:

- ① 根 CA 证书加载。
- ② CA 证书加载。
- ③ 证书作废列表加载。
- ④ 证书作废列表更新等。
- ⑤ 系统配置。
- ⑥ 权限分配。

(3) CA 业务操作员审计

CA 业务操作员的下列操作应被记录:

- ① 证书请求批准。
- ② 证书请求拒绝。
- ③ 证书请求分配。
- ④ 证书作废。

(4) RA 业务操作员审计

RA 业务操作员的下列操作应被记录:

- ① 证书请求批准。

- ② 证书请求拒绝。
- ③ 证书请求分配。
- ④ 证书作废。

23.3 数据备份

数据备份的目的是确保 CA 与 KMC 的关键业务数据在发生灾难性破坏时,系统能够及时和尽可能完整地恢复被破坏的数据。应选择适当的存储备份系统对重要数据进行实时备份、定期备份、增量备份、归档检索与恢复。

不同的应用环境可以有不同的备份方案,但应满足以下基本要求:

- ① 备份要在不中断数据库使用的前提下实施。
- ② 备份方案应符合国家有关信息数据备份的标准要求。
- ③ 备份方案应提供人工和自动备份功能。
- ④ 备份方案应提供实时和定期备份功能。
- ⑤ 备份方案应提供增量备份功能。
- ⑥ 备份方案应提供日志记录功能。

23.4 系统可靠性

CA 与 KMC 必须提供 7×24 小时服务,对影响系统可靠性的主要因素(如网络故障、主机故障、数据库故障和电源故障等)需要采取冗余配置等措施。

1. 网络链路冗余

为保证 CA 的服务,CA 网络对外接口应根据具体情况,可有两条物理上独立的链路,同时考虑交换机、路由器、防火墙的冗余配置。

2. 主机冗余

CA 系统中与关键业务相关的主机、在服务网段和核心网段中的服务器应采用双机热备份或双机备份措施。

3. 数据库冗余

CA 系统的数据库应采用磁盘阵列、磁盘镜像等措施,具备容错和备份能力。

4. 电源冗余

CA 系统应采用高可靠的电源解决方案,并应采用 UPS 为系统提供不间断电源。

23.5 物理安全

1. 物理环境建设

CA 的建筑物及机房建设应按照国家密码管理相关政策要求,并遵照下列标准实施:

- ① GB 9361—88:《计算站场地安全要求》。

- ② GB 2887—89:《计算站场地技术条件》。
- ③ GB 6650—86:《计算机机房用活动地板技术条件》。
- ④ GB 50174—93:《电子计算机机房设计规范》。

2. 对 CA 的分层访问

CA 系统按功能分为 4 个区域,由外到里分别是:公共区、服务区、管理区和核心区。

(1) 公共区

入口之外的区域为公共区。

(2) 服务区

所有进入此区的人员使用身份识别卡刷卡进入。该区的每扇窗户都应安装玻璃破碎报警器。

(3) 管理区

所有进入此区的人员需要同时使用身份识别卡和人体特征鉴别才可以进入,人员进出管理区要有日志记录。所有的房间不应安装窗户,所有的墙体应采用高强度防护墙。

(4) 核心区

所有进入此区的人员需要同时使用身份识别卡和人体特征鉴别才可以进入,人员进出该区要有日志记录。

核心区应为屏蔽机房,应加装高强度的钢制防盗门。所有进出屏蔽室的线路都要采取防电磁泄漏措施。屏蔽效果应符合国家密码管理相关政策要求并达到国家相关标准要求。

(5) 安全监控和配电消防

CA 应设置安全监控室、系统监控室、配电室和消防器材室。

安全监控室是安全管理人员值班的地方,可对整个 CA 的进出人员实行监控,处理日常的安全事件。只有安全管理人员同时使用身份识别卡和人体特征鉴别才可以进入,刷卡离开。

系统监控室是网络管理人员工作的地方,需要同时使用身份识别卡和人体特征鉴别才可以进入,刷卡离开。

配电室是放置所有供电设备的房间,只有相应的授权人员同时使用身份识别卡和人体特征鉴别才可以进入,刷卡离开。

消防器材室是存放消防设备的房间,建议使用身份识别卡进入消防器材室。

3. 门禁和物理侵入报警系统

CA 应设置门禁和物理侵入报警系统。

门禁系统控制各层门的进出。工作人员都需使用身份识别卡或结合人体特征鉴别才能进出,并且进出每一道门都应有时间记录和相关信息提示。

任何非法闯入、非正常手段的开门以及授权人刷卡离开后房内还有非授权的滞留人员,都应触发报警系统。报警系统应明确地指出报警部位。

门禁和物理侵入报警系统应自备有 UPS,并应提供至少 8 小时的供电。

与门禁和物理侵入报警系统配合使用的还应有录像监控系统。对监控区域进行 24 小时不间断的录像。所有的录像资料要根据需要保留一段时间,以备查询。

23.6 人事管理制度

人事管理制度包括人员的可信度鉴别、岗位设置等。

CA 应制定可信人员策略并据此进行人员的可信度鉴别和聘用。可信人员必须接受并通过广泛的背景调查，才能证明他们有能力进行相应关键操作所必需的信任级别。

CA 对人员的教育水平、从业经历、信用情况等方面进行调查，以评估人员的可信度。进行可信人员背景调查必须遵循国家的有关法律、法规和政策。

第24章 运营文件

24.1 CPS

CPS (Certification Practice Statement, 电子认证业务规则) 是电子认证服务机构对所提供的认证及相关业务的全面描述。

按照《中华人民共和国电子签名法》第十九条:“电子认证服务提供者应当制定、公布符合国家有关规定的电子认证业务规则,并向国务院信息产业主管部门备案。电子认证业务规则应当包括责任范围、作业操作规范、信息安全保障措施等事项。”法律规定,CA 必须制定完整的认证业务规则,才能够提供合法的认证业务。缺少 CP/CPS 的 CA,就不是合法的 CA。

CP/CPS 应以文档的形式发布,在文档中说明证书相关的各种情况,而且要对 PKI 的依赖方、订户完全公开,使依赖方和订户能够更安全、更有效地使用 PKI 的服务。

CPS 说明了 CA/PKI 系统的方方面面的信息,即相应 CP 证书的产生、维护、赔偿、法律责任等信息,让使用者(PKI 应用系统)愿意接受其 CP,并将其用于合适的场合。

依据工业和信息化部《电子认证业务规则规范(试行)》办法,电子认证业务规则包括责任范围、作业操作规范和信息安全保障措施等内容,主要由以下几部分组成。

- ① 概括性描述。
- ② 信息发布与信息管理的。
- ③ 身份标识与鉴别。
- ④ 证书生命周期操作要求。
- ⑤ 认证机构设施、管理和操作控制。
- ⑥ 认证系统技术安全控制。
- ⑦ 证书、证书作废列表和在线证书状态协议。
- ⑧ 认证机构审计和其他评估。
- ⑨ 法律责任和其他业务条款。

在“概括性描述”部分,对电子认证业务规则进行概要性表述,给出文档的名称和标识,指出电子认证活动的参与者及证书应用范围,并说明对电子认证业务规则的管理,最后给出电子认证业务规则中使用的定义和缩写。

在“信息发布与信息管理的”部分,描述任何与认证信息发布相关的内容,包括信息库的运营者、运营者的职责、信息发布的频率以及对所发布信息的访问控制等。

在“身份标识与鉴别”部分,描述电子认证服务机构在颁发证书之前,对证书申请者的身份和其他属性进行鉴别的过程,以及标识和鉴别密钥更新请求者和作废请求者的方法。说明命名规则,包括在某些名称中对商标权的承认问题。

在“证书生命周期操作要求”部分,说明在证书生命周期方面对电子认证服务机构及

相关实体的要求，注册机构、订户或其他参与者的要求，在每个子项中可能需要对电子认证服务机构、注册机构、订户或其他参与者予以分别考虑。

在“认证机构设施、管理和操作控制”部分，描述物理环境、操作过程和人员的安全控制。电子认证服务机构使用这些控制手段来安全地实现密钥生成、实体鉴别、证书签发、证书作废、审计和归档等功能。也可定义信息库、注册机构、订户或其他参与者的非技术安全控制。

在“认证系统技术安全控制”部分，阐述电子认证服务机构为保护其密钥和激活数据（如 PIN 码、口令字或手持密钥共享）而采取的安全措施。说明对证书库、订户和其他参与者进行的限制，以保护他们的私钥、私钥激活数据和关键安全参数。描述电子认证服务机构使用的其他技术安全控制手段，用以安全地实现密钥生成、用户鉴别、证书注册、证书作废、审计和归档等功能。技术控制包含生命周期安全控制（包括软件开发环境安全，可信的软件开发方法论）和操作安全控制。

在“证书、证书作废列表和在线证书状态协议”部分，说明证书、证书作废列表和在线证书状态协议的格式，包括描述、版本号 and 扩展项的使用。

在“认证机构审计和其他评估”部分，说明对电子认证服务机构进行审计或评估相关的内容，包括评估所涵盖的主题、评估频率、评估者的资质、被评估者的资质、对问题所采取的措施以及结果的公告等。

在“法律责任和其他业务条款”部分，涵盖了一般性的业务和法律问题。在业务条款中说明不同服务的费用问题，和各参与方为了保证资源维持运营，针对参与方的诉讼和审判提供支付所需承担的财务责任。法律责任条款则与通用的技术协定标题相近，涉及保密、隐私、知识产权、担保及免责等内容。

24.2 CP

根据 X.509 标准，CP（Certificate Policy，证书策略）作为一组规则，表明了证书在特定范围内的、和/或某些具有相同安全需求的应用内的适用程度，也就是说明证书能够用于“安全需求为×××”的应用中。

一个 CA 可以支持签发多种不同等级 CP 的证书，不同等级 CP 的证书用于不同的应用，当然，CA 也可以只支持一种 CP，相当于对证书没有分级，只有一种级别。根 CA 的多个子 CA 可以分别支持不同的 CP。

CP 和 CPS 都说明了 PKI 用户（也就是依赖方）对于证书的可信赖程度，CP 给出了证书的可信赖程度、安全等级。CPS 说明，为了达到相应 CP 的安全等级有什么样的措施和要求，既包括对 CA 的要求，也包括对订户的要求。例如，对高级证书，要求订户只能在电磁辐射屏蔽的机房内使用，否则 CA 不赔偿。

CP 证书策略不包含操作细节，以保持 CP 的长期稳定不变，而操作细节随时间而变化。CP 的设计符合基础设施的思想，应用系统只看到其结果，可以不知道具体操作。CP 与密钥用途截然不同，二者没有联系。密钥用途是从密码运算的角度区分证书用途，证书策略是从安全等级的角度区分证书用途。

CP 标识了证书的安全等级，为 PKI 应用系统提供了使用上的指导和根据。在证书扩

展项中, CP 表示为 OID 的形式, 不同的 CA 公司, 可以定义自己的 OID, 来表示不同的级别。证书的可靠程度由 CPS 保证。例如, CP 和 CPS 关系可以通过一个简单示例来说明: CP 定义的高级证书可用于 100 万人民币以下的电子交易, OID 为 1.3.6.1.4.1.21315.5.1; CPS 措施中, 对于高级证书, 如果出现任何问题, CA 立即无条件赔偿 200 万人民币。

在 CP 的制定上, CA 公司可以不具有自己的 CP, 可以是由第三方制定的。CP 都以文档的形式表现并发布, 在文档中说明了证书相关的各种情况。例如, 如下 CP 设计:

CP1——使用者 DN 允许匿名/假名, 512 位 RSA 密钥, CRL 更新周期 10 天。

CP2——使用者 DN 不允许匿名/加密, 1024 位 RSA 密钥, CRL 更新周期 3 天。

CP3——使用者 DN 不允许匿名/加密, 1024/2048 位 RSA 密钥, CRL 更新周期 1 天。

CP 设计中应该考虑以下问题:

- ① 安全级别覆盖范围, 覆盖从“最简单基本”到“最复杂严格的”安全要求。
- ② 具有可扩展性, 将来可能的各种应用的安全要求应该能够方便地映射到某一级别 CP。
- ③ 一套 CP 中的各个 CP, 其安全等级应有差异, 应用系统才能更好地选择合适的 CP。
- ④ 尽可能只体现安全级别, 不体现具体的操作流程和方法(放在 CPS 中, 可以随着时间变化而修改)。

24.3 RA 管理

RA (Registration Authority, 注册机构) 作为电子认证服务机构授权委托的下属机构, 负责受理证书申请, 包括提交证书申请、审核证书申请、提交作废申请、审核作废申请、提交密钥恢复申请、审核密钥恢复申请、发布审核结果、查询用户、查看用户证书信息、删除用户等功能。

对 RA 的管理应包括系统建设与运维、证书服务等方面, 对于外部建设 RA, 还应包括业务运营和责任及赔偿。

1. 系统建设与运维方面

必须遵循电子认证服务机构的 CPS (电子认证业务规则) 中规定的管理操作要求。采用的 RA 系统 (或产品) 需要经过密码主管部门的评测与认证。物理环境上, 确保 RA 系统位于安全的物理环境, 终端电脑需具备必要的系统运行环境和安全防护措施。对运营人员进行可信雇员调查, 确保其资格、背景、经历符合运营要求, 并保证人员具有适当的知识、技能、素质, 通过培训考核后可以进行相关业务操作。RA 系统日常运行维护上, 应确保数字证书操作终端设备的稳定运行, 并将日常运行情况进行记录, 以备后期审查。系统备份上, 需定期对系统关键数据进行备份, 特别是数据库的自动备份和系统关键数据的自动备份, 并具备完善的系统恢复方案, 当系统发生异常时, 能够快速恢复到正常业务工作状态。

2. 证书服务方面

在接受证书申请和证书更新等各类证书业务时, 应严格依据和遵守电子认证服务机构的 CPS, 对证书申请者的身份进行鉴证, 收取相应的证书申请者鉴证材料和证书申请信息等用户资料, 保证证书申请者的身份信息真实、完整和准确, 不得在用户资料缺失的情况下随意受理证书业务。在受理点存放的证书用户申请资料和证书生产、服务、管理所需的所有原材

料，应使用独立的安全设备存放，确保防潮、防盗、防火等各项安全措施落实到位。在证书申请发放上，RA 向 CA 中心提交证书请求之前，需确认证书申请者的身份已得到鉴证并与证书申请信息相符，确认申请者已认同由电子认证服务机构制定或认可的订户协议，确认证书申请过程中已遵循电子认证服务机构对物理环境、人员和信息安全等要求并遵循所有适用的法律。RA 系统需对系统中的各种操作以日志的形式记载，以方便系统错误分析、风险分析、安全审计等工作。记录日志包括系统运行日志、业务运行日志、操作员操作日志等。

3. 外部建设方面

对于外部建设 RA（即由独立第三方机构管理运维），除了系统建设与运维、证书服务要求外，还包括业务运营和责任及赔偿规范。在业务运营方面，需要规范“证书产品交付及结算，数字证书对账，数字证书结算”等方面。在责任及赔偿方面，涉及“鉴证责任，业务审查责任，赔偿范围，赔偿限额”等。外部建设 RA 需要遵循电子认证服务机构的 CPS 开展业务。

针对人员控制，某 CA 中心（简称 XXCA）CPS 定义如下：

5.3 人员控制

5.3.1 资格、经历和无过失要求

所有的员工与 XXCA 签定保密协议。对于充当可信角色或其他重要角色的人员，必须具备一定的资格，具体要求在人事管理制度中规定。XXCA 要求充当可信角色的人员至少必须具备忠诚、可信赖及工作的热诚度、无影响 CA 运行的其他兼职工作、无同行业重大错误记录、无违法记录等。

5.3.2 背景审查程序

XXCA 与有关的政府部门和调查机构合作，完成对 XXCA 可信员工的背景调查。

所有目前的可信任员工和申请调入的可信任员工都必须书面同意对其进行背景调查。

背景调查分为：基本调查和全面调查。

基本调查包括对工作经历、职业推荐、教育、社会关系方面的调查。

全面调查除包含基本调查项目外还包括对犯罪记录、社会关系和社会安全方面的调查。

调查程序包括：

a) 人事部门负责对应聘人员的个人资料予以确认。提供如下资料：履历、最高学历毕业证书、学位证书、资格证及身份证等相关有效证明。

b) 人事部门通过电话、信函、网络、走访等形式对其提供的材料的真实性进行鉴定。

c) 用人部门通过现场考核、日常观察、情景考验等方式对其考察。

d) 经考核，人事部门和用人部门联合填写《可信雇员调查表》，报主管领导批准后准予上岗。

5.3.3 培训要求

XXCA 对运营人员按照其岗位和角色安排不同的培训。培训有：系统硬件安装与维护、系统软件运行与维护、系统安全、应用软件的运行和维护、CA 中心的运行管理、CA 中心的内部管理、政策和规定及系统备份与恢复等。

对于运营人员，其 CA 的相关知识技能，每年至少要总结一次并由 XXCA 组织培训。技术的进步、系统功能更新或新系统的加入，都需要对相关人员进行培训。

5.3.4 再培训周期和要求

对于充当可信角色或其他重要角色的人员，每年至少接受 XXCA 组织的培训一次。

认证策略调整、系统更新时，应对全体人员进行再培训，以适应新的变化。

5.3.5 工作轮换周期和顺序

对于可替换角色，XXCA 将根据业务的安排进行工作轮换。轮换的周期和顺序视业务的具体情况而定。

5.3.6 对未授权行为的处罚

当 XXCA 员工被怀疑，或者已进行了未授权的操作，例如滥用权利或超出权限使用 XXCA 系统或进行越权操作，

XXCA 得知后将立即对该员工进行工作隔离，随后对该员工的未授权行为进行评估，并根据评估结果对该员工进行相应处罚和采取相应的防范处理措施。情节严重的，依法对其追究相应责任。

5.3.7 独立合约人的要求

对不属于 XXCA 内部的工作人员，但从事 XXCA 有关业务的人员等独立签约者（如注册机构的工作人员），XXCA 的统一要求如下：

- a) 人员档案进行备案管理；
- b) 具有相关业务的工作经验；
- c) 必须接受 XXCA 组织的为期一周的岗前培训。

5.3.8 提供给员工的文档

为使系统正常运行，必须提供给具有权限的相关人员各种文档，包括：

- a) 加密机用户手册；
- b) 机房设备管理办法；
- c) 密码信封打印工具用户手册；
- d) 数字证书运营规范；
- e) 灾难备份和恢复方案；
- f) 目录服务器安装配置手册。

第25章 业务管理

25.1 管理模式

25.1.1 总体框架

1. 业务管理模式总体框架

CA 中心作为第三方电子认证服务机构，是独立的法人企业，将为多个行业的多种应用提供服务，具有很强的通用性和独立性，证书规模至少在几十万以上，投资规模也比较大。CA 中心不仅要满足上级主管部门的监管要求，而且需要通过市场化经营实现自身的生存和发展，具有以下特点：

① 需要快速适应市场需求，提供多种产品服务。CA 中心提供的产品就是数字证书，不同用户或行业领域对证书产品的具体要求可能不同，如证书中内容、有效期、密码算法等。不同用户或行业领域提供的产品可能完全独立，不允许互用，如税务领域证书可能不允许用于社保领域。相同用户或行业领域可能包含多种证书产品。

② 应方便计费收费管理。CA 中心计费类型应包括介质费、证书服务等。应支持多种收费方式，如现场缴费、远程汇款、网上支付等。还应支持优惠促销活动。

③ 应满足主管部门的监管要求。应保存好相关记录，包括业务纸质档案、证书记录、业务申请记录等，且方便查询。

④ 需要为客户提供方便服务。应支持灵活且安全的发证点管理，不同发证点可授权办理不同类型业务。应支持发证点的调整或合并。

CA 中心业务管理模式总体框架如图 25-1 所示。

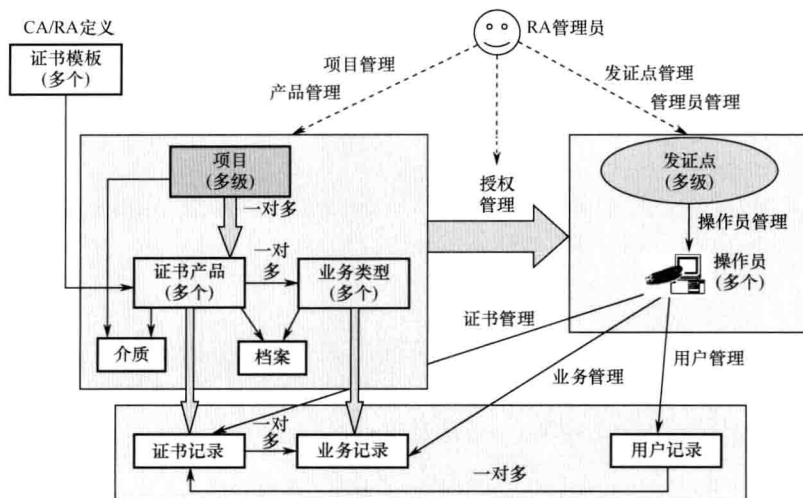


图 25-1 CA 中心业务管理模式总体框架

其中，通过证书模板来快速设计不同的证书产品，证书产品基于证书模板快速生成。业务类型包括申请、作废等，其操作对象为证书产品。通过项目来管理不同的用户或行业领域，基于项目对发证点进行业务授权。

2. 广义 RA 系统

除证书模板功能外，其他业务管理功能应该由 RA 来实现。传统的 RA 并不包括上述业务管理功能，只包括纯证书业务，即证书申请/补办/更新、证书作废、证书冻结/解冻、证书解锁等。为便于说明，本书将传统 RA 系统称为狭义 RA 系统，而将包含计费收费、档案管理、项目及产品管理、发证点管理、用户管理等功能的 RA 系统称为广义 RA 系统。

广义 RA 系统与 CA、KMC、应用系统等关联方的关系模型如图 25-2 所示，其中远程 RA 可以是广义 RA，也可以是狭义 RA。

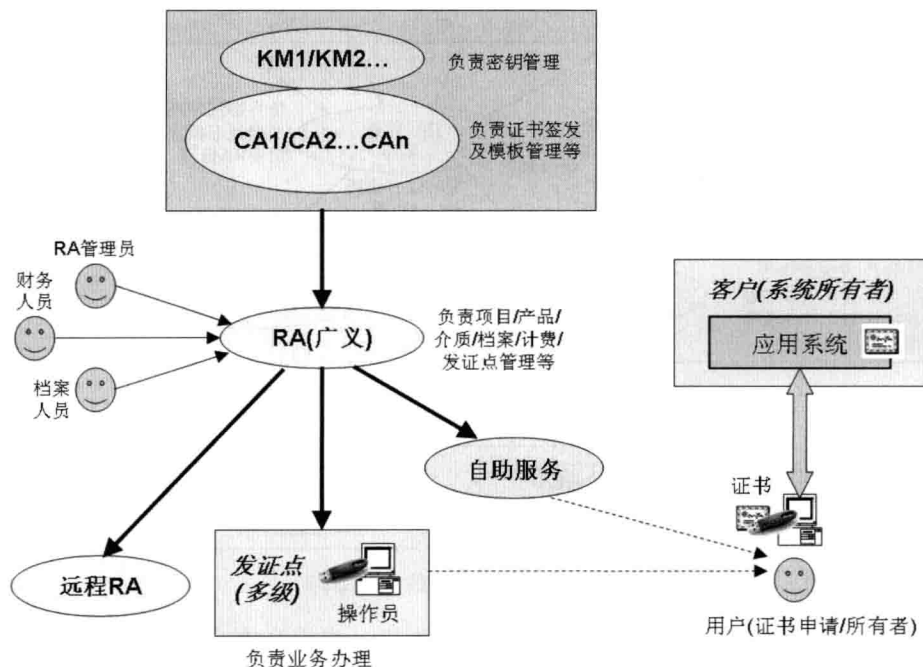


图 25-2 广义 RA 系统关联方模型图

考虑到以下几点原因，CA 中心可能会存在多套 CA 系统和 KMC 系统，不同类型证书可能由不同 CA 签发，而且不同系统可能采用不同厂商产品：

① 不同用户或行业领域的证书产品需要不同的密码算法。如 A 行业可能需要 RSA 1024 和 SHA1 算法，B 行业可能需要 RSA 2048 和 SHA1 算法，C 行业可能需要 SM2 和 SM3 算法。如电子政务证书和电子商务证书可能采用不同算法。

② 密码算法存在安全期，只在一定的时期内是安全的，超过安全期后，该密码算法就会被新的密码算法替代。如 RSA 1024 将被 RSA 2048 或 SM2 替代。

③ 在市场竞争的影响下，有些系统厂商可能会被淘汰。

为适应或屏蔽多个 CA 系统的差异性，需要增加协议转换系统，负责协议转换（通信

及报文)、证书模板管理、RA 管理等功能。其中, RA 管理包括 RA 服务器证书、允许的证书模块及证书数量等。RA 从协议转换系统下载证书模板, 基于证书模块进行管理授权。

为实现多个 RA 情况下的用户统一管理, 需要建设独立的用户管理系统, 负责用户汇总、证书统计和分析、用户分析和挖掘等功能。其中, 当 RA 分多级时, 子 RA 需同时提交用户信息和证书信息, 行业 RA 只需提交证书信息。

扩展上述功能后, CA 系统的系统结构将演变为如图 25-3 所示。

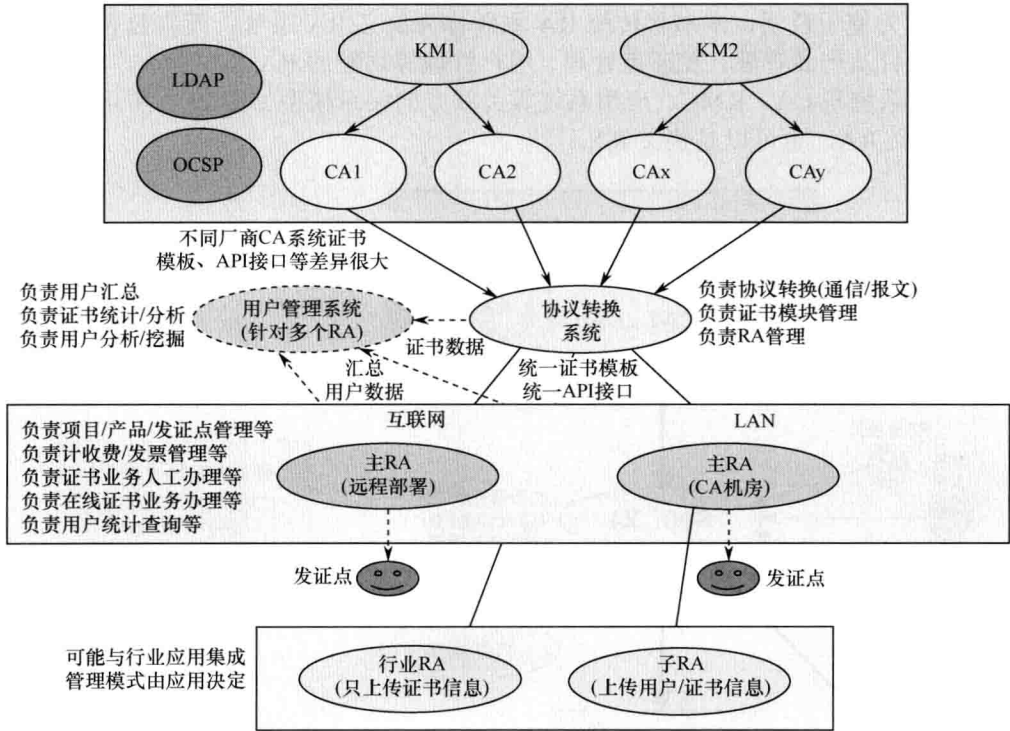


图 25-3 演变后 CA 的系统结构

因广义 RA 系统完成大部分业务管理功能, 为进一步提高客户服务满意度, 需要跟专业的客服系统实现互联互通。广义 RA 系统的系统结构如图 25-4 所示。

25.1.2 具体要求

1. 证书模板

证书模板特指按照 X.509 证书格式框架, 对某类证书格式中的各项内容进行预先定义的一种数据集合。

证书模板基本要求如下:

- ① 证书模板属于技术范畴, 不属于业务范畴。
- ② 证书模板应由 CA 管理, 不允许 RA 修改。
- ③ 应能对证书模板进行授权: 授权给 RA。
- ④ 证书模板应包含运营 CA 系统或证书签发服务器、申请人算法、签名算法等; 不同 CA 系统、不同申请者密钥算法、不同签名算法应使用不同的证书模板。

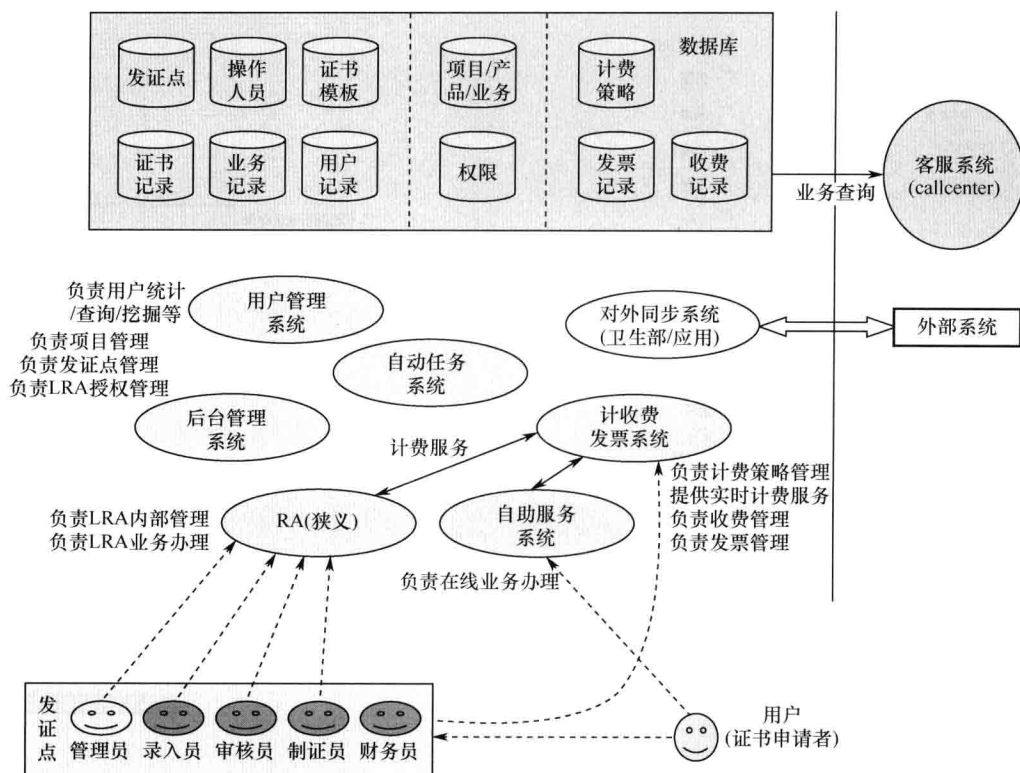


图 25-4 广义 RA 系统的系统结构

图 25-5 和图 25-6 给出了证书模板定义的一个实例，前者为基本项定义，后者为标准扩展项定义。

模板类型	<input checked="" type="radio"/> 单证书 <input type="radio"/> 双证书
密钥产生位置	<input checked="" type="radio"/> 客户端 <input type="radio"/> 密钥管理中心
密钥算法	<input type="text" value="RSA"/>
密钥长度	<input type="text" value="1024"/>
默认有效期	<input type="text" value="365"/> 天
最大有效期	<input type="text" value="3650"/> 天
微软密钥类型	<input type="radio"/> 签名 <input checked="" type="radio"/> 交换
证书发布至目录服务器	<input type="radio"/> 是 <input checked="" type="radio"/> 否
证书发布至文件路径	<input type="radio"/> 是 <input checked="" type="radio"/> 否
证书更新时替换同Key内旧证书	<input type="radio"/> 是 <input checked="" type="radio"/> 否
是否定制CRL发布点	<input type="radio"/> 是 <input checked="" type="radio"/> 否
CRL发布点	<input type="text" value=""/>
是否指定CA签名算法	<input type="radio"/> 是 <input checked="" type="radio"/> 否
CA签名证书主题	<input type="text" value="CN=RSA_DemoCA O=JIT C=CN"/>
CA签名算法	<input type="text" value="sha1RSA"/>

图 25-5 证书模板定义：基本项

<input type="checkbox"/> 颁发密钥标识符	<input type="checkbox"/> 关键
<input type="checkbox"/> 主题密钥标识符	<input type="checkbox"/> 关键
<input type="checkbox"/> 颁发机构信息访问	<input type="checkbox"/> 关键
<input type="checkbox"/> CRL发布点	<input type="checkbox"/> 关键
<input type="checkbox"/> 证书策略	<input type="checkbox"/> 关键
<input type="checkbox"/> 基本限制	<input type="checkbox"/> 关键
<input type="checkbox"/> CA	
证书路径长度约束 <input type="text" value="-1"/>	
<input type="checkbox"/> 密钥用法	<input type="checkbox"/> 关键
	<input type="checkbox"/> 数字签名
	<input type="checkbox"/> 不可否认
	<input type="checkbox"/> 密钥加密
	<input type="checkbox"/> 数据加密
	<input type="checkbox"/> 密钥协商
	<input type="checkbox"/> 证书签名
	<input type="checkbox"/> CRL签名
	<input type="checkbox"/> 只用作加密
	<input type="checkbox"/> 只用作解密
<input checked="" type="checkbox"/> 增强型密钥用法	<input type="checkbox"/> 关键
<input type="checkbox"/> 标准密钥用法	<input type="checkbox"/> 服务端认证
	<input type="checkbox"/> 客户端认证
	<input type="checkbox"/> 代码签名
	<input type="checkbox"/> Email保护
	<input type="checkbox"/> 时间戳
	<input type="checkbox"/> OCSP签名
	<input type="checkbox"/> 智能卡登录
<input type="checkbox"/> 自定义密钥用法	<input type="text" value="OID"/>
<input type="button" value="删除"/>	<input type="button" value="添加自定义增强密钥用法"/>
<input type="checkbox"/> Netscape证书类型	<input type="checkbox"/> 关键
	<input type="checkbox"/> SSL客户
	<input type="checkbox"/> SSL服务
	<input type="checkbox"/> S/MIME
	<input type="checkbox"/> 对象签名
	<input type="checkbox"/> SSLCA
	<input type="checkbox"/> S/MIMECA
	<input type="checkbox"/> 对象签名CA
<input type="checkbox"/> 策略映射	<input type="checkbox"/> 关键
	<input type="checkbox"/> 证书申请时必须指定值
<input type="checkbox"/> 策略限制	<input type="checkbox"/> 关键
	<input type="checkbox"/> 证书申请时必须指定值
<input type="checkbox"/> 主题备用名称	<input type="checkbox"/> 关键
	<input type="checkbox"/> IP地址
	<input type="checkbox"/> 电子邮件地址
	<input type="checkbox"/> 用户甄别名
	<input type="checkbox"/> 其它名称
	<input type="checkbox"/> 域名
	<input type="checkbox"/> 统一资源定位
	<input type="checkbox"/> 证书申请时必须指定值
<input type="checkbox"/> 证书模板名称	<input type="checkbox"/> 关键
	<input checked="" type="radio"/> 域控制器(DomainController)
	<input type="radio"/> 智能卡登录用户(SmartCardLogonUser)
	<input type="radio"/> 自定义模板名称

图 25-6 证书模板定义：标准扩展项

2. 用户管理

用户特指证书申请者或证书所有者。证书中 Subject 与证书所有者不一定一致。
用户管理基本要求如下：

① 用户分为 2 类：个人和单位。单位可以是法人或政府机关，也可以是法人或政府机关内的某个部门。

② 应支持独立的用户管理功能，以方便客户分析、挖掘、跟踪等。

③ 每个用户可拥有多个证书。

④ 用户信息与证书中的信息有交叉。可以设定映射关系，避免重复录入。

⑤ 用户证书到期前应提供提醒机制：短信、邮箱等。

⑥ 用户身份核验时，企业只须提供营业执照、税务登记证或组织机构代码证之一即可；由于企业名称变更后营业执照或税务登记证编号可能会发生编号（组织机构代码不变），因此企业用户的唯一标识应慎重考虑。

3. 项目管理

项目是客户（系统所有者）或同类用户（证书所有者）的统称。客户与用户关系如图 25-7 所示，如国税局为客户，网上报税系统的所有者为国税局，用户即证书所有者为报税企业。

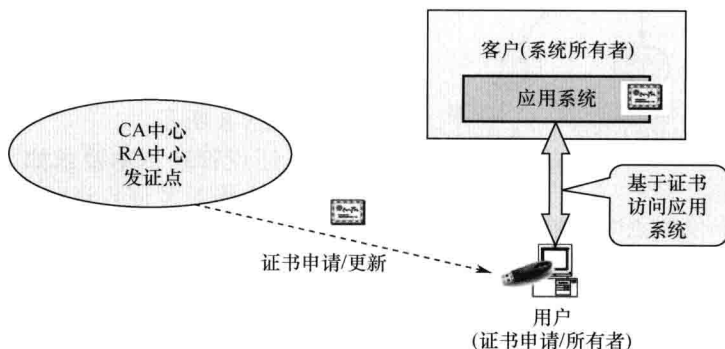


图 25-7 客户与用户关系图

项目管理基本要求如下：

- ① 项目允许多级。
- ② 项目应包括多个产品。

4. 产品管理

产品特指数字证书。

产品管理基本要求如下：

- ① 产品应隶属于某个项目。
- ② 产品应基于证书模板建立。
- ③ 多个产品可以基于相同的证书模板。
- ④ 应能设置某类产品所允许的介质类型，允许多个。
- ⑤ 产品应能设置待录入信息的缺省值，如 DN 项中 C/S/L/O/OU、有效期、介质类型等。
- ⑥ 产品应能设置应提交的业务资料清单（依据 CPS）。

5. 介质管理

介质特指用户私钥及证书载体，如 Key 或 IC 卡等。

介质管理基本要求如下：

- ① 可给发证点、项目或项目中证书类型设定所允许的介质类型。
- ② 介质应具有唯一的硬件序列号，在证书申请时系统应能自动记录该序列号。
- ③ 当已有介质重复申请证书时，应自动提示。

④ 对于不同密码算法的介质,访问接口应统一管理。如可采用 CSP (需扩展)或国密接口统一访问 SM2 算法或 RSA 算法的介质等。

⑤ 用户证书更新时可能需要自己输入介质口令,发证点可考虑配置用户密码键盘。

6. 发证点管理 (LRA)

发证点特指能为证书申请者提供纯证书业务 (证书申请/补办/更新、证书作废、证书冻结/解冻、证书解锁等) 和支撑类业务 (计费收费、档案管理、项目及产品管理、发证点管理、用户管理等) 且独立经营的业务单元。

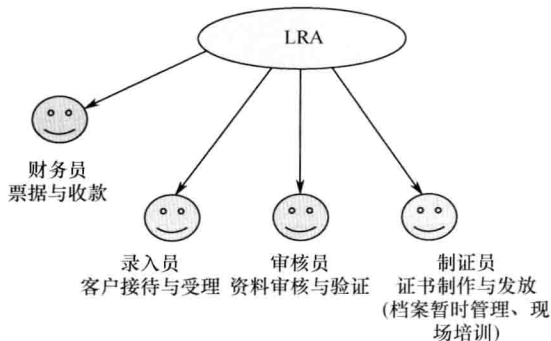


图 25-8 发证点岗位设置

从岗位设置考虑,发证点的主要业务工作可分为客户接待与受理、证书资料审核与验证、证书制作与发放 (档案暂时管理、现场培训)、票据与收款等 4 类。为控制风险,证书受理、资料审核与验证不应由同一个人承担,其他岗位可一人多岗。发证点岗位设置如图 25-8 所示。

发证点管理基本要求如下:

① 应对对发证点进行控制或授权。具体包括:发证数量 (包括总发证数量、某项目总发证数量、某项目某类证书总发证数量等)、能发哪些项目的证书、能发项目中的哪类证书、设定管理员、是否允许自主管理 (如创建下级发证点、增加操作员、变更操作员角色、给操作员授权等)、某类证书允许使用的介质等。

② 发证点自身安全控制。具体包括:应支持多级;上级发证点业务人员具有下级发证点的对应权限;发证点人员至少分 2 类 (管理员和操作员。操作员应分为多类角色:录入员、审核员、制证员、财务人员,管理员可对操作员进行管理);发证点所有人员应使用数字证书进行身份认证;发证点操作员应负责资料收取、信息录入、业务审核、费用收取等职责;发证点管理员应负责资料归档及上报、费用汇总及结算等职责;发证点应不用部署任何系统,直接通过浏览器进行业务操作;管理员对操作员进行授权 (如哪些项目、哪类证书等)。

③ 应支持发证点的调整或合并,保证针对原有证书的各种业务不受影响。具体包括:某些发证点的上级发证点发生变更;某些发证点合并。

④ 发证点应能验证用户信息。

⑤ 跨发证点业务办理。具体包括:不同发证点可能属于不同的利益集团;当用户在发证点 A 申请证书,在发证点 B 更新证书时,属于跨发证点业务办理,应提供支持。

7. 业务类型管理

证书业务类型主要包括:证书申请、证书作废、证书解冻、证书冻结、证书解锁、证书更新 (信息不变)、证书变更 (信息变化) 等。

业务类型管理基本要求如下:

① 应记录每笔证书业务的详细信息,包括时间、类型、费用、操作对象、操作者、签名等。业务记录不仅包括发证点业务操作,也包括用户自助服务业务操作。

② 当发证点操作员的业务操作失误时 (如 DN 项录入错误后制证成功),应允许重新制证 (原证书应作废),并增加标记,不影响证书或业务统计查询、计费收费。

③ 针对证书更新时存在的 RSA 更新为 SM2、单证书更新为双证书等情况,可增加不同

的证书更新业务类型来实现,如证书更新 I (RSA 到 SM2)、证书更新 II (单证到双证) 等。

④ 证书业务流程主要包括以下环节:受理、审核、收费、制证、绑定、发票、结算、归档等,如图 25-9 所示。

⑤ 应支持批量申请和批量下载功能。如基于 Excel 文件批量提交、按照特定顺序批量下载(如 Excel 文件顺序)等。

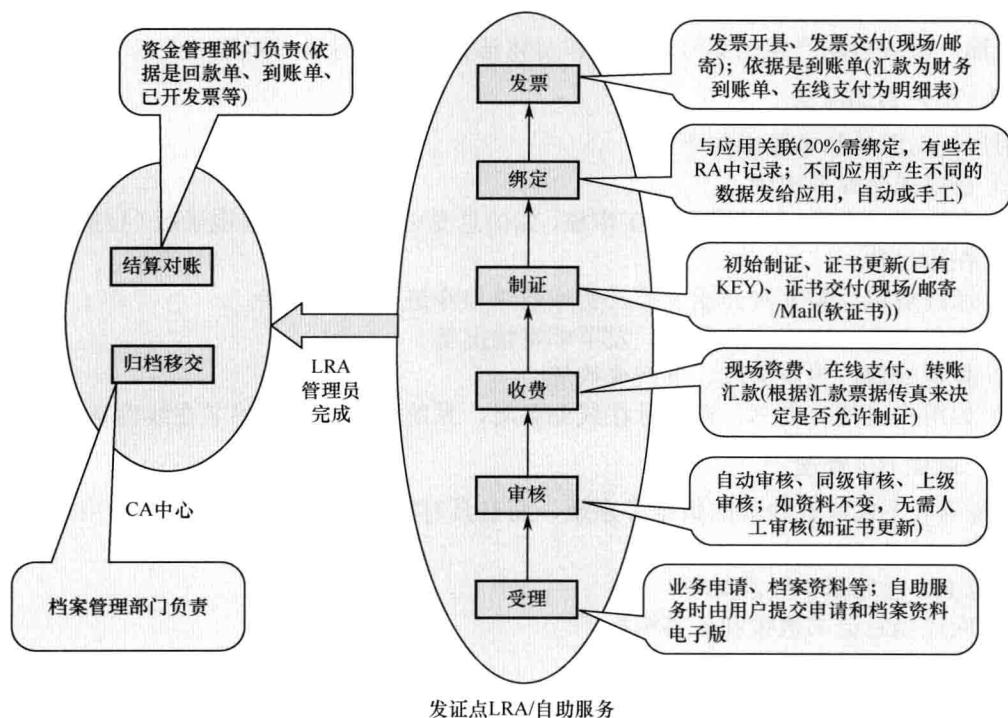


图 25-9 证书业务流程

8. 计费收费

证书业务收费有两种模式:用户自主付费和客户承担费用。客户是指应用系统所有者,如国税局,用户是证书申请者,如报税企业。

用户自主付费模式下,计费收费管理基本要求如下:

- ① 发证点需按照计费策略,实时计算费用并收取。
- ② 业务费用应包括介质费、证书服务费、增值服务等。
- ③ 应支持用户以多种方式缴费:现场缴费(现金/支票)、远程汇款(邮局汇款/银行转账)、网上支付(个人网银/企业网银/第三方支付)等。
- ④ 支持多种优惠促销政策。
- ⑤ 支持证书变更时,新证书有效期可能小于一个完整计费周期时的计费规则。
- ⑥ 支持退费业务。
- ⑦ 应支持多种发票送达方式:现场交付、事后自取、事后邮寄(需录入邮寄地址)等。

9. 档案管理

档案特指证书业务中需要用户提交的各种书面资料。

档案管理基本要求如下：

① 可以给项目设定该项目允许的档案清单，也可以直接给项目中某类证书设定该证书允许的档案清单，或者给某类证书的某类业务设定该业务允许的档案清单。如营业执照复印件、经办人身份证复印件、证书申请书、企业介绍信等。

② 证书业务所需档案清单，应依据 CPS 设定。

③ 为方便档案管理，在进行业务处理时，操作员选择或录入用户实际提交的各种书面资料名称，系统自动产生档案号，由操作员将该档案号书写到书面资料上。

10. 用户自助服务

用户自助服务基本要求如下：

① 在线提交业务申请。

② 在线证书更新：可能需要人工审核，如信息变更，需上传资料电子版（扫描件/照片）。

③ 在线付款。

④ 远程解锁：需要对介质主控密钥或解锁口令进行统一管理。

⑤ 业务查询：申请业务状态、发票邮寄情况等。

⑥ 异常处理：更新失败、下载失败等。

⑦ 如用户已拥有证书，在进行在线业务时，系统可通过用户签名在线确认用户身份。

11. 远程 RA 管理

远程 RA 不仅具有独立的数据库系统，而且具有独立的发证点/项目/产品/用户/计收费管理功能。

远程 RA 管理基本要求如下：

① 应控制总证书数量和证书模板。

② 用户数据是否向上同步。

③ 远程 RA 应通过服务器证书保证安全性。

④ 应能控制远程 RA 管理员。

12. 证书追溯（生命周期展现）

证书追溯是指对与某用户、某证书或某介质关联的所有证书进行查询并显示出来。

证书追溯机制具体要求如下：

① 针对某用户，应能展示所有历史证书，即使用户名称发生变更。

② 针对某证书，应能展示所有更新的历史证书，即使密钥发生变化。

③ 针对某介质，应能展示所有历史证书，即使用户发生变化；包括返修 KEY。

25.1.3 管理模式示例

1. 示例 A

图 25-10 给出了一种管理模式的实现形式，其中证书模板中的属性项可自定义，功能比较强大；证书模板支持授权管理；支持独立的用户管理功能；支持双证书和两种模式单证书（用户端产生和 KMC 产生密钥对）等。

2. 示例 B

图 25-11 给出了一种管理模式的实现形式，其中证书模板中的属性项不允许自定义，

证书模板不支持授权管理，不支持独立的用户管理功能，只支持双证书等。

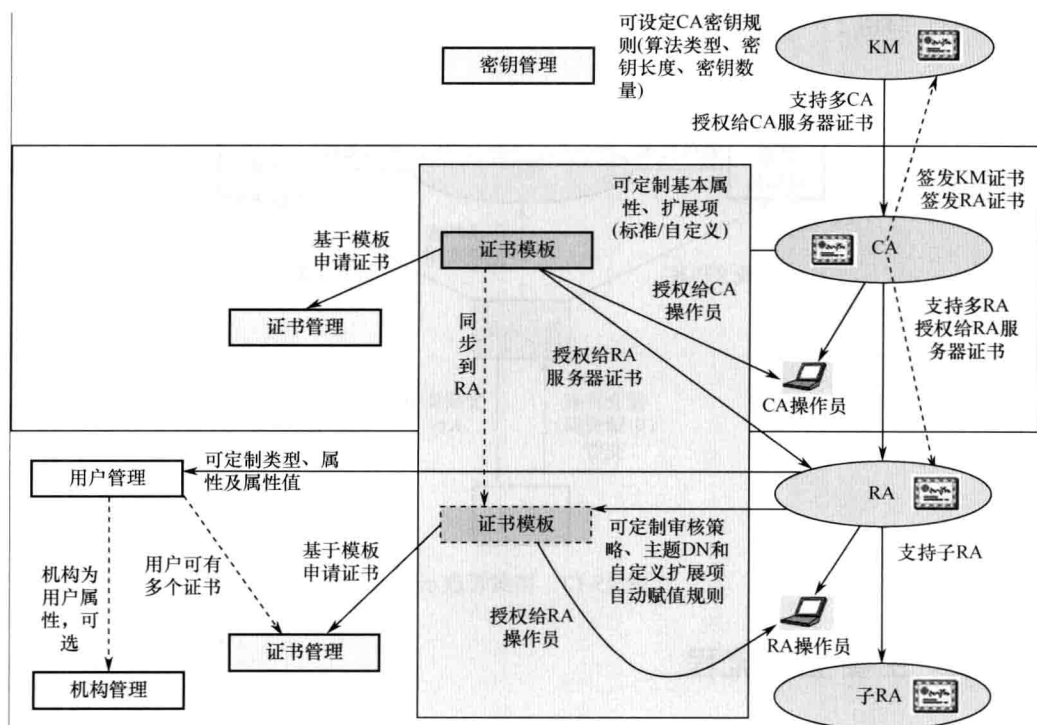


图 25-10 管理模式示例 A

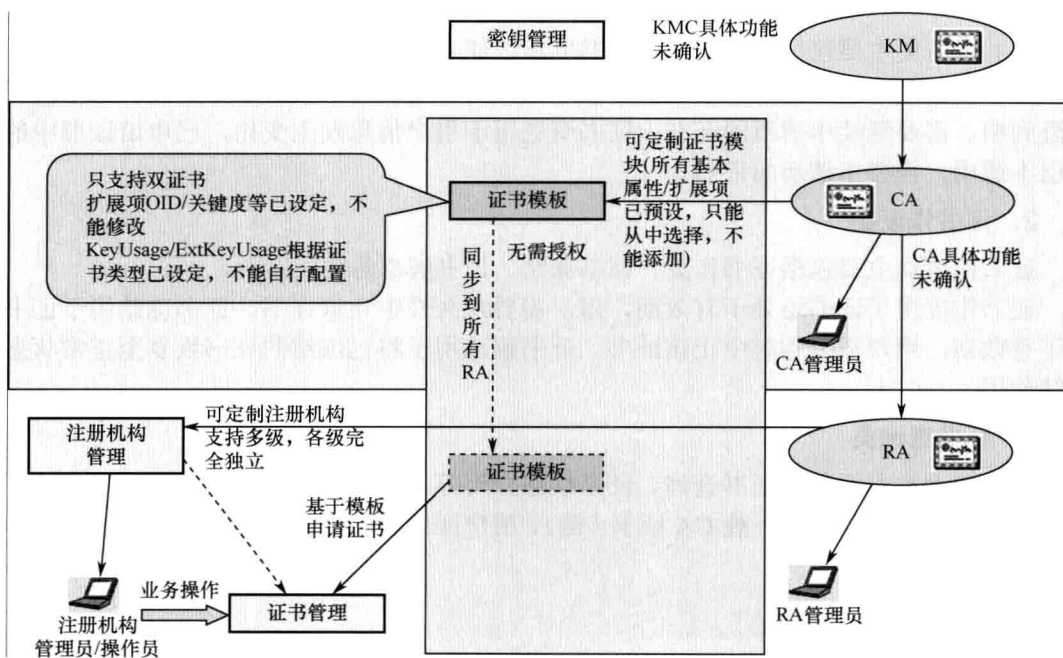


图 25-11 管理模式示例 B

3. 示例 C

图 25-12 给出了一种档案管理的实现形式，其中同时支持项目管理、档案管理、收费管理等。

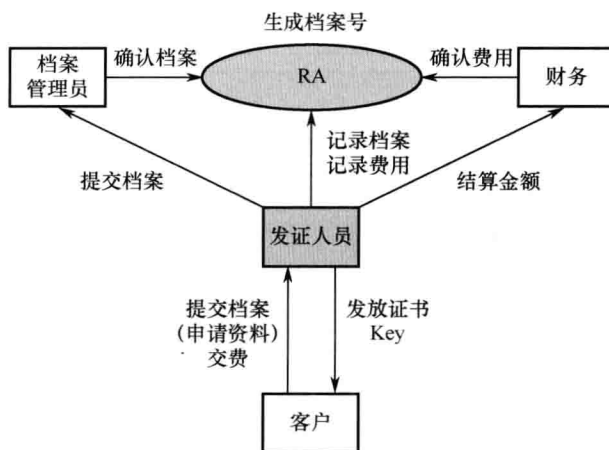


图 25-12 档案管理示例

25.2 主要业务流程

CA 中心面向用户（证书申请者）提供的业务办理类型可分为三大类。

1. 证书申请类

证书申请类主要包括证书申请、证书更新、证书变更等。

证书申请用于用户首次申请证书。证书更新用于用户信息没有发生变化，已申请证书临近到期，需要继续申请新的证书。证书变更用于用户信息发生变化，已申请证书中的内容已不适用，需要申请新的证书。

2. 证书作废类

证书作废类主要包括证书作废、证书冻结、证书解冻等。

证书作废用于证书还处于有效期，用户需要永久性中止该证书。证书冻结用于证书还处于有效期，用户需要临时中止该证书。证书解冻用于将已冻结的证书恢复至正常状态，继续使用。

3. 证书查询类

证书查询类主要包括证书查询、证书状态查询等。

证书查询用于查询并下载 CA 证书（链）、用户证书等。证书状态查询主要采用 OCSP、CRL 等方式。

25.2.1 证书申请类

证书申请类的业务流程如图 25-13 所示。

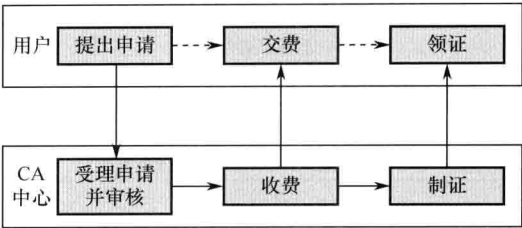


图 25-13 证书申请类业务流程

25.2.1.1 证书申请

1. 提出申请

用户（证书申请者）如实填写“数字证书申请表”，并认真阅读“数字证书服务协议”，按要求在申请表和服务协议上签名或盖章（个人证书申请应签名，机构证书申请应加盖公章）。个人和机构数字证书申请表样例分别参见表 25-1 和表 25-2。数字证书服务协议样例参见表 25-3。

按要求准备相关资料后，向 CA 中心提交证书申请。

个人证书申请须提交的资料主要包括：个人数字证书申请表、数字证书服务协议、申请人证件复印件及原件、代理人证件复印件及原件等。

机构证书申请须提交的资料主要包括：机构数字证书申请表、数字证书服务协议、机构证件复印件及原件（工商营业执照、组织机构代码证等）、经办人证件复印件及原件等。

表 25-1 个人数字证书申请表样例

证书申请信息					
申请日期		证书数量		证书期限	
证书种类	<input type="checkbox"/> 个人普通证书 <input type="checkbox"/> 个人高级证书 <input type="checkbox"/> 安全邮件证书 <input type="checkbox"/> 其他种类证书，请注明：				
业务类型	<input type="checkbox"/> 新申请 <input type="checkbox"/> 更新 <input type="checkbox"/> 作废				
证书 DN（仅更新或作废时填写）					
申请人信息					
姓名（中文）				姓名（拼音）	
证件类型	<input type="checkbox"/> 身份证 <input type="checkbox"/> 户口簿 <input type="checkbox"/> 护照 <input type="checkbox"/> 军人身份证 <input type="checkbox"/> 武警身份证 <input type="checkbox"/> 港澳通行证 <input type="checkbox"/> 其他，请注明：				
证件号码					
电话		传真		电子邮件	
联系地址				邮政编码	
申请人声明					
本人承诺以上所填写的申请资料真实、有效。本人已认真阅读并同意遵守《数字证书服务协议》、《电子认证业务规则（CPS）》中规定的相关义务。					
申请人签名				日期	
以下信息由 CA 中心填写					

(续表)

证书申请信息					
录入员				录入日期	
审核员		审核意见	<input type="checkbox"/> 审核通过 <input type="checkbox"/> 审核不通过	审核日期	
收费员				收费日期	
制证员				制证日期	
备注				CA 中心盖章	

表 25-2 机构数字证书申请表样例

证书申请信息					
申请日期		证书数量		证书期限	
证书种类	<input type="checkbox"/> 机构普通证书 <input type="checkbox"/> 机构高级证书 <input type="checkbox"/> 服务器证书 <input type="checkbox"/> 设备证书 <input type="checkbox"/> 安全邮件证书 <input type="checkbox"/> 代码签名证书 <input type="checkbox"/> RA 管理员证书 <input type="checkbox"/> 其他种类证书, 请注明:				
业务类型	<input type="checkbox"/> 新申请 <input type="checkbox"/> 更新 <input type="checkbox"/> 作废		证书 DN (仅更新或作废时填写)		
申请机构信息					
机构名称					
英文/拼音简称					
机构证件类型	<input type="checkbox"/> 企业营业执照 <input type="checkbox"/> 组织机构代码证 <input type="checkbox"/> 其他, 请注明:				
机构证件号码					
网站域名或 IP 地址 (仅申请服务器证书时填写)					
机构经办人信息					
经办人姓名					
经办人证件类型	<input type="checkbox"/> 身份证 <input type="checkbox"/> 户口簿 <input type="checkbox"/> 护照 <input type="checkbox"/> 军人身份证 <input type="checkbox"/> 武警身份证 <input type="checkbox"/> 港澳通行证 <input type="checkbox"/> 其他, 请注明:				
经办人证件号码					
电话		传真		电子邮件	
联系地址				邮政编码	
申请机构声明					
本机构承诺以上所填写的申请资料真实、有效。本机构已认真阅读并同意遵守《数字证书服务协议》、《电子认证业务规则 (CPS)》中规定的相关义务。					
申请机构盖章				日期	
以下信息由 CA 中心填写					
录入员				录入日期	
审核员		审核意见	<input type="checkbox"/> 审核通过 <input type="checkbox"/> 审核不通过	审核日期	
收费员				收费日期	
制证员				制证日期	
备注				CA 中心盖章	

表 25-3 数字证书服务协议样例

数字证书是电子认证服务机构签发的包含数字证书使用者身份信息和公开密钥的电子文件。某 CA 中心（以下简称 XYZCA）是工业和信息化部批准的电子认证服务机构和国家密码管理局批准的电子政务电子认证服务机构，遵照《中华人民共和国电子签名法》为用户提供数字证书相关的电子认证服务。

本协议中的用户指数字证书持有人以及申请使用数字证书的实体。

为明确各方权利和义务，XYZCA 和用户就数字证书的申请和使用等事宜达成以下协议，共同遵守执行。

第一条 申请

1. 用户在申请数字证书时，应提供真实、完整和准确的信息及证明材料。如因故意或过失未向 XYZCA 或其设立的注册机构提供真实、完整和准确的信息，导致 XYZCA 签发证书错误，造成相关各方损失的，由用户承担相关责任。

2. XYZCA 设立的注册机构作为证书业务受理单位和服务支持单位，负责用户的信息录入、身份审核和证书制作工作。用户在申请数字证书时应遵照注册机构的规程办理手续。

3. XYZCA 设立的注册机构应完全遵守 XYZCA 安全操作流程进行用户身份审核和证书制作。

4. XYZCA 积极响应各注册机构发出的证书申请请求，及时为通过审核的用户签发证书。如果由于设备或网络故障而导致签发数字证书错误、延迟、中断或者无法签发，XYZCA 不承担任何赔偿责任。

5. 用户在获得数字证书时应及时验证此证书所匹配的信息，如无异议则视为接受证书。

第二条 使用

6. XYZCA 签发的数字证书用于网络上的用户身份标识、数字签名验证及密钥分配，各应用系统可根据需要对其用途进行定义，但不包括涉及违反国家法律、法规或危害国家安全的用途。

7. 用户应确保其应用系统能为数字证书提供安全的应用环境，若因网络、主机、操作系统或其他软硬件环境等存在安全漏洞，由此导致的安全事故及相关后果，XYZCA 不承担责任。

8. 用户应当妥善保管 XYZCA 签发的数字证书、私钥及保护密码，不得泄露或交付他人。如用户保管不善导致数字证书遭盗用、冒用、伪造或者篡改，用户应当自行承担相关责任。

9. 数字证书对应的私钥为用户本身访问和使用，用户对使用数字证书的行为负责。所有使用数字证书在网上交易和网上作业中的活动均视为用户所为，因此而产生的相关后果应当由用户自行承担。

10. 数字证书一律不得转让、转借或转用。因转让、转借或转用而产生的相关后果应当由用户自行承担。

11. XYZCA 承诺，在现有的技术条件下，由 XYZCA 签发的数字证书不会被伪造、篡改。如果发生数字证书被篡改、伪造，经确认确属 XYZCA 责任，XYZCA 承担赔偿责任。赔偿方法参照《XYZCA 电子认证业务规则》。

第三条 更新

12. 数字证书的有效期限自证书受理之日起计算。若在数字证书有效期到期后，用户仍需继续使用数字证书，必须在数字证书到期前一个月内向 XYZCA 设立的注册机构提出数字证书更新请求。否则，证书到期将自动失效，XYZCA 对此不承担责任。

13. 因技术需要，XYZCA 有权要求用户及时更新数字证书。用户在收到更新通知后，应在规定的期限内到 XYZCA 设立的注册机构更新证书。若用户逾期没有更新证书，因此而产生的相关后果应当由用户自行负责。

第四条 作废

14. 如遇数字证书私钥泄露丢失、证书中的信息发生重大变更或用户不希望继续使用数字证书的情况，用户应当立即到 XYZCA 设立的注册机构申请作废证书。作废手续遵循各注册机构的规定。XYZCA 在接到作废申请后，在 24 小时内作废用户的数字证书。用户应当承担在证书作废之前所有因使用数字证书而造成的责任。

15. 如果用户主体资格灭失（如企业注销等），法定责任人应携带相关证明文件及原数字证书，向注册机构请求作废用户证书。相关责任人应当承担其数字证书在作废前所有使用数字证书而造成的相关后果。

16. 对于下列情形之一，XYZCA 有权主动作废所签发的证书：

① 用户申请证书时，提供不真实信息；

- ② 证书对应的私钥泄露或出现其他证书的安全性得不到保证的情况；
- ③ 用户不能履行或违反了相关法律、法规和本协议所规定的责任和义务；
- ④ 法律、法规规定的其他情形。

第五条 其他

17. XYZCA 不对由于意外事件或其他不可抗力事件而导致暂停或终止全部或部分证书服务承担任何责任。

18. 本协议条款可由 XYZCA 随时更新，XYZCA 会通过网站 www.xyzca.com 进行通知和公布，更新后的协议一旦公布即替代原来的协议条款。用户如果不接受修改后的协议，可于通知发布之日起十五日内，向 XYZCA 设立的注册机构提出作废证书的申请。如果逾期没有提出异议，则视为同意接受修订后的协议。

19. 本协议与 XYZCA 网站上公布的《XYZCA 电子认证业务规则》共同构成关于数字证书的完整协议。

20. 本协议的解释适用中华人民共和国法律。若用户与 XYZCA 之间发生任何纠纷或争议，首先应友好协商解决，协商不成的，双方同意将纠纷或争议提交×××法院管辖。

用户确认已经认真阅读并完全理解本协议中的各项规定，用户在申请表上签名盖章即表明接受本协议的约束，本协议即时生效。

2. 受理申请并审核

CA 中心录入员负责内容检查和信息录入。如果检查不合格，则提示申请人补充完善；如果检查合格，则将证书申请信息录入 RA 系统。检查内容主要包括：

- ① 申请书是否填写完整和规范。
- ② 申请书和服务协议是否有个人签名或加盖机构公章。
- ③ 携带资料是否齐全。
- ④ 申请书填写内容是否与资料吻合。

CA 中心审核员负责信息审核。如果审核不合格，则在 RA 系统中标记审核不通过、记录不通过理由，并告知申请人原因、返还申请资料；如果审核合格，则在 RA 系统中标记审核通过。审核内容主要包括：

- ① 申请人是否符合证书发放要求。
- ② 相关证件是否真实有效。
- ③ 申请人的信用情况，是否有不良记录。

CA 中心也可以提供 Web 服务，允许用户通过互联网进行证书申请，录入相关信息后，携带资料到现场进行业务办理。

3. 收费

CA 中心收费员负责费用收取。按计费规定收取费用后，给申请人出具收费票据，并在 RA 系统中记录收费情况。

证书申请费用主要包括证书介质费、证书服务费、增值服务等。选用不同型号的证书介质，则证书介质费不同。不同证书类型、不同的证书有效期，其证书服务费不同。增值服务可包括电子印章、短信提醒等。

CA 中心也可以提供 Web 服务，允许用户通过网上支付方式付费。

4. 制证与领证

CA 中心制证员负责证书的制作和发放。通过 RA 系统完成密钥对生成和证书下载，证书和私钥均存储在证书介质中（如 USB Key、IC 卡等），并打印密码信封（用于访问证书

介质的口令)。

制证完毕后,制证员将证书介质和密码信封交给申请人。

25.2.1.2 证书更新

与证书申请相比,证书更新有以下不同。

1. 用户信息已经审核通过

用户在首次申请证书时,已经提交过纸质资料并审核通过。由于用户信息没有发生变化,因此证书更新时,无需重复提交纸质资料,信息审核环节可以简化(申请人的信用情况可能会发生变化,可能需要再次审核)。

如果CA中心提供Web服务,则用户就可以通过Web服务远程提交证书更新申请,无需到现场进行手工办理。因为可以通过用户已有证书进行身份认证,实现远程确认用户的合法身份。

2. 已经拥有证书介质

通常情况下,证书介质可以复用,即证书到期后,使用原有证书介质可以申请新的证书。

如果CA中心提供Web服务,则用户可以通过Web服务远程更新证书到原来的证书介质中。当然,前提是用户支付证书更新费,并经CA中心审核通过。

如果CA中心同时提供Web服务,允许用户远程提交证书更新申请、网上付款、远程更新证书,则证书更新业务流程就可以完全实现在线办理,从而提高效率。

25.2.1.3 证书变更

由于用户信息已发生变化,因此仍需要严格的信息审核。

证书变更业务流程与证书申请基本相同。

25.2.2 证书作废类

证书作废类的业务流程如图25-14所示。

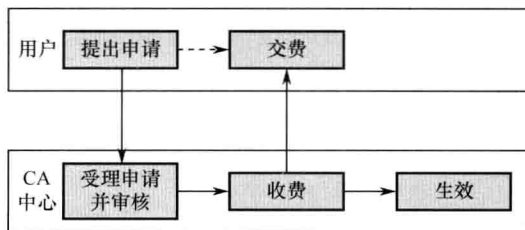


图 25-14 证书作废类业务流程

25.2.2.1 证书作废

当证书仍处于有效期但发生以下原因时,该证书应该作废,不再允许使用:

- ① 证书持有者已改变状态(如信息变更、岗位调整、证件失效等),无权使用证书。
- ② 证书持有者的密钥已遭破坏或证书介质丢失,或者怀疑密钥泄露,为避免遭受损失。
- ③ 证书持有者本人不想再使用该证书。
- ④ 证书签发者CA的密钥已被破坏或到期失效。

⑤ 证书签发者 CA 决定作废该证书等。

如发生上述前三项原因，证书持有者应主动发出证书作废申请。

1. 提出申请

用户（证书作废申请者）如实填写“证书作废申请表”，并认真阅读“数字证书服务协议”，按要求在申请表和服务协议上签名或盖章（个人证书作废申请应签名，机构证书作废申请应加盖公章）。证书作废申请书中需要注明作废理由。个人和机构数字证书作废申请表样例分别参见表 25-1 和表 25-2，数字证书服务协议样例参见表 25-3。

按要求准备相关资料后，向 CA 中心提交证书申请。提交的资料与证书申请类似。

2. 受理申请并审核

CA 中心录入员负责内容检查和信息录入。如果检查不合格，则提示申请人补充完善；如果检查合格，则将作废申请信息录入 RA 系统。检查内容与证书申请类似。

CA 中心审核员负责信息审核。如果审核不合格，则在 RA 系统中标记审核不通过、记录不通过理由，并告知申请人原因、返还申请资料；如果审核合格，则在 RA 系统中标记审核通过。审核内容与证书申请类似。

CA 中心也可以提供 Web 服务，允许用户通过互联网进行证书作废申请，录入相关信息后，携带资料到现场进行业务办理。

3. 收费

CA 中心收费员负责费用收取。按计费规定收取费用后，给申请人出具收费票据，并在 RA 系统中记录收费情况。

CA 中心也可以提供 Web 服务，允许用户通过网上支付方式付费。

4. 生效

审核通过并按要求交费后，CA 中心将及时把该证书状态修改为作废，并通过 CRL 定期发布出去。通过 OCSP 可以实时查询到该证书状态已经处于作废状态。

25.2.2.2 证书冻结与证书解冻

证书冻结、证书解冻业务流程与证书作废流程基本相同，这里不再赘述。

25.2.3 证书查询类

25.2.3.1 证书查询

主要用于查询并下载 CA 证书（链）和用户证书，通常采用 HTTP 或 LDAP 方式。

1. HTTP 方式

由于 CA 证书（链）数量很少，通常以文件形式存在，通过 Web 方式就可以直接下载，无需查询。图 25-15 给出了 HTTP 方式下 CA 证书（链）的查询下载界面示例图。

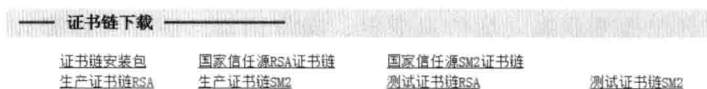


图 25-15 HTTP 方式下 CA 证书（链）的查询下载界面示例

由于用户证书数量很多，通常保存在数据库中，需要通过 Web 方式查询后才能下载。图 25-16 给出了 HTTP 方式用户证书的查询下载界面示例图。

证书状态	证书类型	姓名/名称(CN)	单位(O)	部门(OU)	城市(L)	省份(S)	起始时间	终止时间	操作
已作废	个人	bjramgr222					20101015000000	20121014000000	下载 查看
已作废	个人	bjramgr333					20101015000000	20121014000000	下载 查看
已冻结	个人	bjramake					20101018000000	20121017000000	下载 查看
有效	个人	bjramgr444					20101018000000	20121017000000	下载 查看
有效	个人	bjramgr555					20101018000000	20121017000000	下载 查看
有效	个人	bjramgr666					20101018000000	20121017000000	下载 查看

图 25-16 HTTP 方式下用户证书的查询下载界面示例

2. LDAP 方式

LDAP 方式下，CA 证书（链）和用户证书的存储方式相同，均以条目形式存在。需要采用专用的 LDAP 软件工具或 API 方式，才能对证书进行查询和下载。图 25-17 给出了 Foxmail 6.5 中使用 LDAP 方式进行证书查询的界面示例。

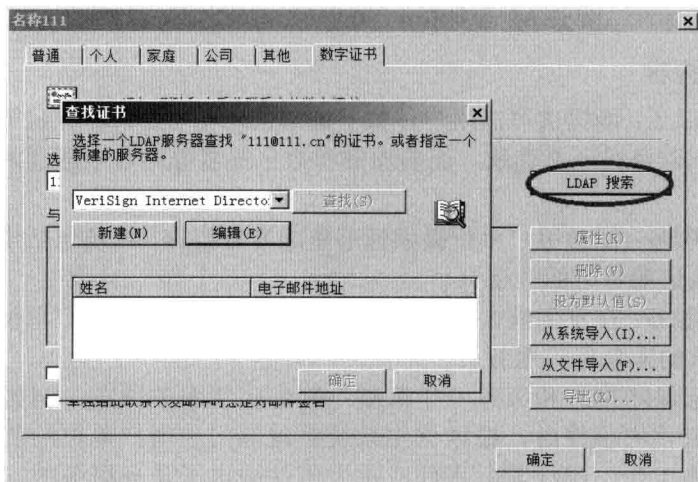


图 25-17 Foxmail 6.5 中 LDAP 证书查询界面

25.2.3.2 证书状态查询

证书状态查询主要采用 OCSP、CRL 等方式。

OCSP 方式下，需要专用的 OCSP 查询工具或 API 方式才能进行证书状态查询。

CRL 方式下，可通过 HTTP 或 LDAP 方式获得。采用 HTTP 方式时，CRL 以文件形式存在，通过 Web 方式就可以直接下载，无需查询。采用 LDAP 方式时，CRL 以条目形式存在，需要采用专用的 LDAP 软件工具或 API 方式，才能对 CRL 进行查询和下载。

25.3 客户服务

鉴于数字证书领域的专业性，在用户使用数字证书过程中可能会涉及多种多样的业务问题和技术问题，需要配置专门的坐席并基于知识库对用户进行全方位客户服务。客户服务与发证点和用户之间的关系如图 25-18 所示。

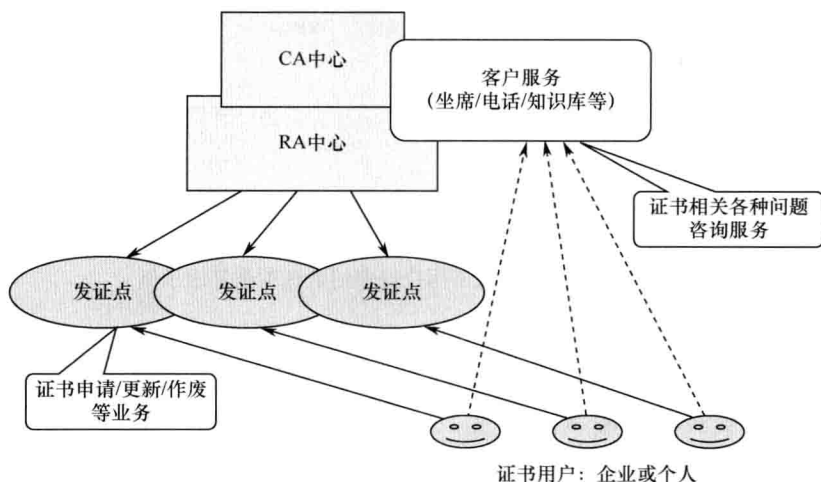


图 25-18 客户服务与发证点和用户之间的关系

1. 客户服务内容

① 使用帮助服务：能提供解决用户在数字证书的申请、更新和解锁等环节中遇到的各类业务办理问题，以及在证书登录、证书加密和数字签名等环节中遇到的各类证书应用问题的帮助服务。

② 咨询培训服务：能对用户单位提供证书集成方案咨询，以及包括电子认证服务相关政策法规、技术认证服务方面的培训。

③ 应急保障服务：在发生重大事件或特殊应急事件时，根据用户单位的要求而临时提供的超越常规要求的应急保障服务。

应该建立应急保障措施和应急工作机制，制定应急服务预案。例如，在卫生信息系统出现紧急事件或重大事件时，根据卫生系统管理部门和用户单位的要求，及时提供现场方式的应急保障工作，确保紧急事件或重大事件中电子认证服务的安全性、可靠性和有效性。

④ 应用集成支持服务：能提供针对各行业信息系统的电子认证安全需求分析、电子认证法律法规、技术体系的咨询，设计满足业务要求的电子认证及电子签名服务方案；应提供面向多种应用环境的证书应用接口程序供应用系统集成和调用。

2. 客户服务分级

客户服务应该分级管理，通常可分为 3 级：

① 紧急服务：直接影响应用系统关键任务运行，并导致业务工作停顿的故障所需要的维修服务。

② 常规服务：系统自身发生故障，影响到安全系统正常运行，但对业务系统影响较小，所需要的维修服务。

③ 计划服务：定期检修服务。

不同客户服务级别，所要求的服务时间、响应时间、解决时限等也不同。表 25-4 给出了一个实例。

表 25-4 客户服务级别要求实例

服务级别	服务时间	响应时间	解决时限	需求程度
紧急服务	7×24 小时	2 小时	24 小时	非常关键、紧急
常规服务	工作日	4 小时	1~3 个工作日	重要但不紧急
计划服务	工作日	一周	1~7 个工作日	重要但可延缓数日

第26章 资质申请

26.1 电子认证服务使用密码许可证

26.1.1 政策法规要点

1. 《电子签名法》（2004 年版）

《电子签名法》（2004 年版）规定：

第十六条 电子签名需要第三方认证的，由依法设立电子认证服务提供者提供认证服务。

第十七条 提供电子认证服务，应当具备下列条件：

- （一）具有与提供电子认证服务相适应的专业技术人员和管理人员；
- （二）具有与提供电子认证服务相适应的资金和经营场所；
- （三）具有符合国家安全标准的技术和设备；
- （四）具有国家密码管理机构同意使用密码的证明文件；
- （五）法律、行政法规规定的其他条件。

2. 《电子认证服务密码管理办法》（2009 年版）

《电子认证服务密码管理办法》（2009 年版，国家密码管理局公告第 17 号）规定：

第二条 国家密码管理局对电子认证服务提供者使用密码的行为实施监督管理。

省、自治区、直辖市密码管理机构依据本办法承担有关监督管理工作。

第三条 提供电子认证服务，应当依据本办法申请《电子认证服务使用密码许可证》。

第四条 采用密码技术为社会公众提供第三方电子认证服务的系统（以下称电子认证服务系统）使用商用密码。

电子认证服务系统应当由具有商用密码产品生产资质的单位承建。

第七条 申请《电子认证服务使用密码许可证》应当在电子认证服务系统建设完成后，向所在地的省、自治区、直辖市密码管理机构提交下列材料：

- （一）《电子认证服务使用密码许可证申请表》；
- （二）企业法人营业执照或者企业名称预先核准通知书的复印件；
- （三）电子认证服务系统安全性审查相关技术材料，包括建设工作总结报告、技术工作总结报告、安全性设计报告、安全管理策略和规范报告、用户手册和测试说明；
- （四）电子认证服务系统互联互通测试相关技术材料；
- （五）电子认证服务系统物理环境符合电磁屏蔽、消防安全有关要求的证明文件；
- （六）电子认证服务系统使用的信息安全产品符合有关法律规定的证明文件。

第十条 《电子认证服务使用密码许可证》载明下列内容：

- （一）许可证编号；
- （二）电子认证服务提供者名称；
- （三）许可证有效期限；
- （四）发证机关和发证日期。

《电子认证服务使用密码许可证》有效期为五年。

26.1.2 申请流程

1. 办理流程

《电子认证服务使用密码许可证》具体办理流程如图 26-1 所示。

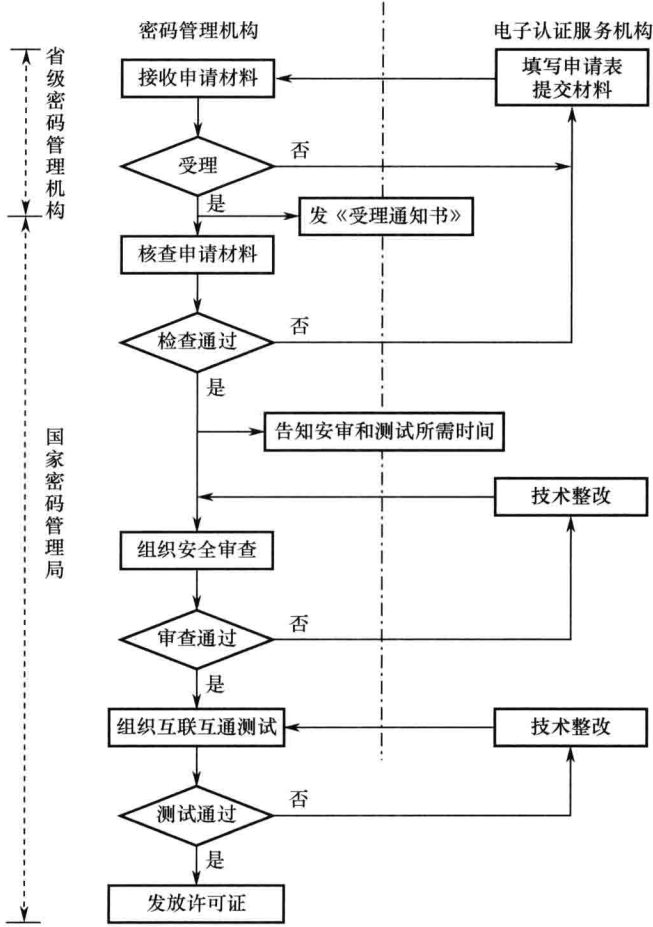


图 26-1 《电子认证服务使用密码许可证》办理流程

2. 申请表

《电子认证服务使用密码许可证》申请表如表 26-1 所示。

表 26-1 《电子认证服务使用密码许可证》申请表

单位名称				原许可证号	
详细地址				邮政编码	
法人代表				电话	
单位负责人				电话	
技术负责人				电话	
申请联系人	姓名			电话	
	EMAIL			传真	
成立时间		注册资本		经济类型	
股权（资本）结构					
CA/KM 系统承建单位					
密钥管理系统所属单位			密钥管理系统主管部门		
申请单位	省级密码管理机构 受理意见： 年 月 日		负责人签字（公章） 年 月 日		
法人代表					
签字（公章）					

26.2 电子认证服务许可证

26.2.1 政策法规要点

1. 《电子签名法》（2004 年版）

《电子签名法》（2004 年版）规定：

第十六条 电子签名需要第三方认证的，由依法设立的电子认证服务提供者提供认证服务。

第十八条 从事电子认证服务，应当向国务院信息产业主管部门提出申请，并提交符合本法第十七条规定条件的相关材料。国务院信息产业主管部门接到申请后经依法审查，征求国务院商务主管部门等有关部门的意见后，自接到申请之日起四十五日内作出许可或者不予许可的决定。予以许可的，颁发电子认证许可证书；不予许可的，应当书面通知申请人并告知理由。

申请人应当持电子认证许可证书依法向工商行政管理部门办理企业登记手续。

取得认证资格的电子认证服务提供者，应当按照国务院信息产业主管部门的规定在互联网上公布其名称、许可证号等信息。

2. 《电子认证服务管理办法》（2009 年版）

《电子认证服务管理办法》（2009 年版）规定：

第二条 本办法所称电子认证服务,是指为电子签名相关各方提供真实性、可靠性验证的活动。

本办法所称电子认证服务提供者,是指为需要第三方认证的电子签名提供认证服务的机构(以下称为“电子认证服务机构”)。

向社会公众提供服务的电子认证服务机构应当依法设立。

第四条 中华人民共和国工业和信息化部(以下简称“工业和信息化部”)依法对电子认证服务机构和电子认证服务实施监督管理。

第五条 电子认证服务机构应当具备下列条件:

(一)具有独立的企业法人资格。

(二)具有与提供电子认证服务相适应的人员。从事电子认证服务的专业技术人员、运营管理人员、安全管理人员和客户服务人员不少于三十名,并且应当符合相应岗位技能要求。

(三)注册资本不低于人民币三千万元。

(四)具有固定的经营场所和满足电子认证服务要求的物理环境。

(五)具有符合国家有关安全标准的技术和设备。

(六)具有国家密码管理机构同意使用密码的证明文件。

(七)法律、行政法规规定的其他条件。

第六条 申请电子认证服务许可的,应当向工业和信息化部提交下列材料:

(一)书面申请。

(二)人员证明。

(三)资金证明(经依法审计的近三年的财务会计报告,新成立公司的验资报告)。

(四)经营场所证明。

(五)国家有关认证检测机构出具的技术、设备、物理环境符合国家有关安全标准的凭证。

(六)国家密码管理机构同意使用密码的证明文件。

第十条 工业和信息化部应当自接到申请之日起四十五日内作出准予许可或者不予许可的书面决定。不予许可的,应当书面通知申请人并说明理由;准予许可的,颁发《电子认证服务许可证》,并公布下列信息:

(一)《电子认证服务许可证》编号。

(二)电子认证服务机构名称。

(三)发证机关和发证日期。

电子认证服务许可相关信息发生变更的,工业和信息化部应当及时公布。

《电子认证服务许可证》的有效期为五年。

26.2.2 申请流程

1. 办理流程

《电子认证服务许可证》具体办理流程如图 26-2 所示。

2. 提交材料

申请从事电子认证服务的企业需提交符合法定形式的以下材料(一式两份,正式装订):

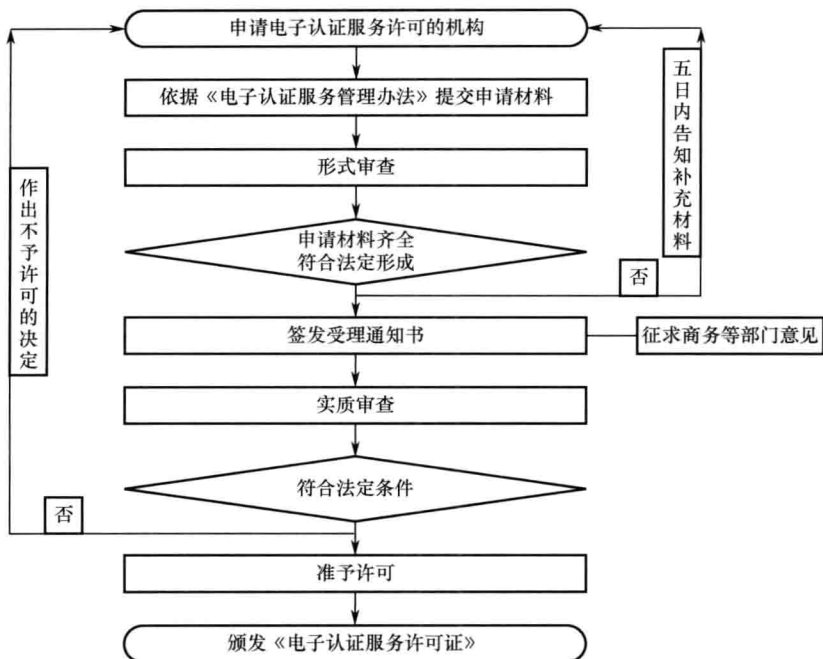


图 26-2 《电子认证服务许可证》办理流程

一、电子认证服务行政许可申请表；

二、企业法人资格证明（企业法人营业执照副本复印件）；

三、人员证明（企业的专业技术人员、运营管理人员、安全管理人员和客户管理人员聘用合同复印件）；

四、资金证明（经依法审计的近三年的企业财务会计报告复印件，或新成立企业的验资报告复印件）；

五、经营场所证明（经营场所产权证明文件复印件或有效的房屋租赁协议复印件）；

六、国家有关认证检测机构出具的技术、设备、物理环境符合国家有关安全标准的凭证：

（一）有关安全设备和认证系统在有效期内的经安全认证和准予销售的凭证复印件；

（二）技术体系审查报告的复印件；

（三）消防工程、核心屏蔽机房、运营机房经主管部门或国家认可的检测机构验收报告复印件。

七、申请单位公司章程复印件；

八、国家密码管理机构同意使用密码的证明文件复印件；

九、申请单位认为有必要提供或工业和信息化部信息安全协调司要求提供的其他材料。

26.3 电子政务电子认证服务管理

《电子政务电子认证服务管理办法》（2009 年版）规定：

第二条 本办法所称电子政务电子认证服务（以下简称认证服务），是指电子认证服务

机构采用密码技术，通过数字证书，为各级政务部门开展社会管理、公共服务等政务活动提供的电子认证服务。

电子政务电子认证服务包括面向政务部门的电子政务电子认证服务和面向企事业单位、社会团体、社会公众的电子政务电子认证服务。

第四条 国家密码管理局负责全国电子政务电子认证服务活动的监督管理工作。

各省、自治区、直辖市和中央国家机关有关部委密码管理部门（以下简称省部密码管理部门）按照国家密码管理局的统一要求，负责本地区本部门电子政务电子认证服务活动的监督管理工作。

第十五条 电子认证服务机构应当具备以下条件：

- （一）事业法人或者取得电子认证服务许可的国有控股企业法人；
- （二）具有经国家密码管理局批准的电子政务电子认证基础设施；
- （三）具有与从事认证服务相适应的专业技术人员、运行维护人员、安全管理人员和服务人员；
- （四）具有与认证服务相适应的运行服务、应用支持和安全保障等机制；
- （五）法律法规规定的其他条件。

第十六条 面向企事业单位、社会团体、社会公众提供服务的电子认证服务机构，应当取得国务院信息产业主管部门颁发的《电子认证服务许可证》。

第十七条 国家密码管理局组织开展认证服务能力评估，发布《电子政务电子认证服务机构目录》。

第十八条 电子认证服务机构申请进行认证服务能力评估时，应当向所属省部密码管理部门提交以下材料：

- （一）书面申请；
- （二）事业单位应当提交组织机构代码证书、法人证书以及其他相关证明文件的复印件；企业应当提交组织机构代码证书、营业执照、《电子认证服务许可证》、公司章程以及其他相关证明文件的复印件；
- （三）电子政务电子认证基础设施经国家密码管理局批准的证明材料的复印件；
- （四）专业技术人员、运行维护人员、安全管理人员和服务人员的资格证明材料；
- （五）本机构的电子认证服务业务规则；
- （六）运行服务、应用支持和安全保障等机制的相关材料；
- （七）国家密码管理局规定的其他材料。

26.4 卫生系统电子认证服务管理

26.4.1 政策法规要点

《卫生系统电子认证服务管理办法》（2009年版）规定：

第三条 本办法所称电子认证服务，是指电子认证服务机构按照卫生系统电子认证服务体系相关技术标准和服务规范，为使用方提供数字证书的颁发、更新、作废、密码解锁、密钥恢复以及证书应用等服务，从而满足卫生信息系统在身份认证、授权管理、责任认定等方面的信息安全需求。

第五条 卫生部信息化工作领导小组负责全国卫生系统电子认证服务体系的建设和管理工作，负责组织制定卫生系统电子认证服务相关技术标准和服务规范。各省级卫生行政部门信息化工作领导小组负责指导、推进本辖区卫生信息系统开展安全、规范的电子认证服务应用。

第六条 电子认证服务机构面向卫生系统提供电子认证服务应当具备以下必要条件：

（一）取得工业和信息化部颁发的《电子认证服务许可证》；

（二）符合《电子政务电子认证体系建设总体规划》（国密局联字〔2007〕2号）中关于电子认证体系建设的相关要求；

（三）具有符合《卫生系统电子认证服务规范》、《卫生系统数字证书格式规范》、《卫生系统数字证书介质技术规范》、《卫生系统数字证书应用集成规范》和《卫生系统数字证书服务管理平台接入规范》等的电子认证服务体系；

（四）符合法律、行政法规规定的其他条件。

第七条 卫生部信息化工作领导小组办公室负责选择卫生部及其直属单位的电子认证服务机构。各省级卫生行政部门信息化工作领导小组办公室负责选择本辖区的电子认证服务机构。

第八条 各省级卫生行政部门信息化工作领导小组办公室选定电子认证服务机构后，应当将服务机构名单及其相关材料向卫生部信息化工作领导小组办公室上报，并要求电子认证服务机构在开展服务前接入卫生部数字证书服务管理平台。

第九条 鼓励各地卫生系统数字证书应用单位优先选择已接入卫生部数字证书服务管理平台的电子认证服务机构提供服务。

26.4.2 接入流程

各省级卫生行政部门选定的电子认证服务机构在开展服务前，要按照工作程序接入卫生部数字证书服务管理平台。具体接入流程如下。

1. 申请阶段

按照《办法》要求，拟纳入卫生系统电子认证服务体系的电子认证服务机构须向省级卫生行政部门提交申请和相关资料。具体资料包括：

- ① 公司营业执照复印件（加盖企业公章）；
- ② 工业和信息化部颁发的《电子认证服务许可证》复印件（加盖企业公章）；
- ③ 企业国有资产控股证明文件（加盖企业公章）；
- ④ 《××省（区、市）卫生系统电子认证服务保障方案》，内容包括服务内容（证书签发、发放、更新、作废、解锁、培训等）、服务流程、签证措施、信息安全保障、服务模式（现场窗口、远程在线服务）、服务热线及服务人员等详细的运行服务方案；
- ⑤ 《卫生系统电子认证服务机构接入申请表》。

2. 省级卫生行政部门资料初审、评估阶段

省级卫生行政部门要及时对电子认证服务机构提交的申请材料进行审核，并对机构情况进行全面评估。通过审核评估的电子认证服务机构，要将其相关资料和申请表（加盖省级卫生行政部门公章）报卫生部办公厅。

3. 卫生部复审、测试阶段

卫生部办公厅接收申请资料后，将进行申请资料审核和系统接入测试，时间不超过 1 个月。系统接入测试步骤如下：

① 第一步，电子认证服务机构从卫生部统计信息中心领取测试规范进行自测。

② 第二步，自测完成后，卫生部统计信息中心按照测试规范，对电子认证服务机构的电子认证系统、证书介质、证书格式、证书应用接口等进行测试，测试通过后报卫生部办公厅备案。

4. 接入、公布结果

通过测试的电子认证服务机构将其电子认证服务系统正式接入卫生部数字证书服务管理平台。接入成功后，卫生部办公厅将测试结果通报省级卫生行政部门，并通过卫生部网站公布已成功接入的电子认证服务机构名单。

第七部分

PKI 之法规与标准

第27章 国内法规

27.1 电子签名法

（中华人民共和国主席令 第十八号，第十届全国人民代表大会常务委员会第十一次会议于2004年8月28日通过，自2005年4月1日起施行）

第一章 总则

第一条 为了规范电子签名行为，确立电子签名的法律效力，维护有关各方的合法权益，制定本法。

第二条 本法所称电子签名，是指数据电文中以电子形式所含、所附用于识别签名人身份并表明签名人认可其中内容的数据。

本法所称数据电文，是指以电子、光学、磁或者类似手段生成、发送、接收或者储存的信息。

第三条 民事活动中的合同或者其他文件、单证等文书，当事人可以约定使用或者不使用电子签名、数据电文。

当事人约定使用电子签名、数据电文的文书，不得仅因为其采用电子签名、数据电文的形式而否定其法律效力。

前款规定不适用下列文书：

- （一）涉及婚姻、收养、继承等人身关系的；
- （二）涉及土地、房屋等不动产权益转让的；
- （三）涉及停止供水、供热、供气、供电等公用事业服务的；
- （四）法律、行政法规规定的不适用电子文书的其他情形。

第二章 数据电文

第四条 能够有形地表现所载内容，并可以随时调取查用的数据电文，视为符合法律、法规要求的书面形式。

第五条 符合下列条件的数据电文，视为满足法律、法规规定的原件形式要求：

- （一）能够有效地表现所载内容并可供随时调取查用；
- （二）能够可靠地保证自最终形成时起，内容保持完整、未被更改。但是，在数据电文上增加背书以及数据交换、储存和显示过程中发生的形式变化不影响数据电文的完整性。

第六条 符合下列条件的数据电文，视为满足法律、法规规定的文件保存要求：

- （一）能够有效地表现所载内容并可供随时调取查用；
- （二）数据电文的格式与其生成、发送或者接收时的格式相同，或者格式不相同但是能够准确表现原来生成、发送或者接收的内容；

(三) 能够识别数据电文的发件人、收件人以及发送、接收的时间。

第七条 数据电文不得仅因为其是以电子、光学、磁或者类似手段生成、发送、接收或者储存的而被拒绝作为证据使用。

第八条 审查数据电文作为证据的真实性,应当考虑以下因素:

- (一) 生成、储存或者传递数据电文方法的可靠性;
- (二) 保持内容完整性方法的可靠性;
- (三) 用以鉴别发件人方法的可靠性;
- (四) 其他相关因素。

第九条 数据电文有下列情形之一的,视为发件人发送:

- (一) 经发件人授权发送的;
- (二) 发件人的信息系统自动发送的;
- (三) 收件人按照发件人认可的方法对数据电文进行验证后结果相符的。

当事人对前款规定的事项另有约定的,从其约定。

第十条 法律、行政法规规定或者当事人约定数据电文需要确认收讫的,应当确认收讫。发件人收到收件人的收讫确认时,数据电文视为已经收到。

第十一条 数据电文进入发件人控制之外的某个信息系统的时间,视为该数据电文的发送时间。

收件人指定特定系统接收数据电文的,数据电文进入该特定系统的时间,视为该数据电文的接收时间;未指定特定系统的,数据电文进入收件人的任何系统的首次时间,视为该数据电文的接收时间。

当事人对数据电文的发送时间、接收时间另有约定的,从其约定。

第十二条 发件人的主营业地为数据电文的发送地点,收件人的主营业地为数据电文的接收地点。没有主营业地的,其经常居住地为发送或者接收地点。

当事人对数据电文的发送地点、接收地点另有约定的,从其约定。

第三章 电子签名与认证

第十三条 电子签名同时符合下列条件的,视为可靠的电子签名:

- (一) 电子签名制作数据用于电子签名时,属于电子签名人专有;
- (二) 签署时电子签名制作数据仅由电子签名人控制;
- (三) 签署后对电子签名的任何改动能够被发现;
- (四) 签署后对数据电文内容和形式的任何改动能够被发现。

当事人也可以选择使用符合其约定的可靠条件的电子签名。

第十四条 可靠的电子签名与手写签名或者盖章具有同等的法律效力。

第十五条 电子签名人应当妥善保管电子签名制作数据。电子签名人知悉电子签名制作数据已经失密或者可能已经失密时,应当及时告知有关各方,并终止使用该电子签名制作数据。

第十六条 电子签名需要第三方认证的,由依法设立的电子认证服务提供者提供认证服务。

第十七条 提供电子认证服务,应当具备下列条件:

- (一) 具有与提供电子认证服务相适应的专业技术人员和管理人员;
- (二) 具有与提供电子认证服务相适应的资金和经营场所;
- (三) 具有符合国家安全标准的技术和设备;
- (四) 具有国家密码管理机构同意使用密码的证明文件;
- (五) 法律、行政法规规定的其他条件。

第十八条 从事电子认证服务,应当向国务院信息产业主管部门提出申请,并提交符合本法第十七条规定条件的相关材料。国务院信息产业主管部门接到申请后经依法审查,征求国务院商务主管部门等有关部门的意见后,自接到申请之日起四十五日内作出许可或者不予许可的决定。予以许可的,颁发电子认证许可证书;不予许可的,应当书面通知申请人并告知理由。

申请人应当持电子认证许可证书依法向工商行政管理部门办理企业登记手续。

取得认证资格的电子认证服务提供者,应当按照国务院信息产业主管部门的规定在互联网上公布其名称、许可证号等信息。

第十九条 电子认证服务提供者应当制定、公布符合国家有关规定的电子认证业务规则,并向国务院信息产业主管部门备案。

电子认证业务规则应当包括责任范围、作业操作规范、信息安全保障措施等事项。

第二十条 电子签名人向电子认证服务提供者申请电子签名认证证书,应当提供真实、完整和准确的信息。

电子认证服务提供者收到电子签名认证证书申请后,应当对申请人的身份进行查验,并对有关材料进行审查。

第二十一条 电子认证服务提供者签发的电子签名认证证书应当准确无误,并应当载明下列内容:

- (一) 电子认证服务提供者名称;
- (二) 证书持有人名称;
- (三) 证书序列号;
- (四) 证书有效期;
- (五) 证书持有人的电子签名验证数据;
- (六) 电子认证服务提供者的电子签名;
- (七) 国务院信息产业主管部门规定的其他内容。

第二十二条 电子认证服务提供者应当保证电子签名认证证书内容在有效期内完整、准确,并保证电子签名依赖方能够证实或者了解电子签名认证证书所载内容及其他有关事项。

第二十三条 电子认证服务提供者拟暂停或者终止电子认证服务的,应当在暂停或者终止服务九十日前,就业务承接及其他有关事项通知有关各方。

电子认证服务提供者拟暂停或者终止电子认证服务的,应当在暂停或者终止服务六十日前向国务院信息产业主管部门报告,并与其他电子认证服务提供者就业务承接进行协商,作出妥善安排。

电子认证服务提供者未能就业务承接事项与其他电子认证服务提供者达成协议的,应当申请国务院信息产业主管部门安排其他电子认证服务提供者承接其业务。

电子认证服务提供者被依法吊销电子认证许可证书的,其业务承接事项的处理按照国

务院信息产业主管部门的规定执行。

第二十四条 电子认证服务提供者应当妥善保存与认证相关的信息，信息保存期限至少为电子签名认证证书失效后五年。

第二十五条 国务院信息产业主管部门依照本法制定电子认证服务业的具体管理办法，对电子认证服务提供者依法实施监督管理。

第二十六条 经国务院信息产业主管部门根据有关协议或者对等原则核准后，中华人民共和国境外的电子认证服务提供者在境外签发的电子签名认证证书与依照本法设立的电子认证服务提供者签发的电子签名认证证书具有同等的法律效力。

第四章 法律责任

第二十七条 电子签名人知悉电子签名制作数据已经失密或者可能已经失密未及时告知有关各方、并终止使用电子签名制作数据，未向电子认证服务提供者提供真实、完整和准确的信息，或者有其他过错，给电子签名依赖方、电子认证服务提供者造成损失的，承担赔偿责任。

第二十八条 电子签名人或者电子签名依赖方因依据电子认证服务提供者提供的电子签名认证服务从事民事活动遭受损失，电子认证服务提供者不能证明自己无过错的，承担赔偿责任。

第二十九条 未经许可提供电子认证服务的，由国务院信息产业主管部门责令停止违法行为；有违法所得的，没收违法所得；违法所得三十万元以上的，处违法所得一倍以上三倍以下的罚款；没有违法所得或者违法所得不足三十万元的，处十万元以上三十万元以下的罚款。

第三十条 电子认证服务提供者暂停或者终止电子认证服务，未在暂停或者终止服务六十日前向国务院信息产业主管部门报告的，由国务院信息产业主管部门对其直接负责的主管人员处一万元以上五万元以下的罚款。

第三十一条 电子认证服务提供者不遵守认证业务规则、未妥善保存与认证相关的信息，或者有其他违法行为的，由国务院信息产业主管部门责令限期改正；逾期未改正的，吊销电子认证许可证书，其直接负责的主管人员和其他直接责任人员十年内不得从事电子认证服务。吊销电子认证许可证书的，应当予以公告并通知工商行政管理部门。

第三十二条 伪造、冒用、盗用他人的电子签名，构成犯罪的，依法追究刑事责任；给他人造成损失的，依法承担民事责任。

第三十三条 依照本法负责电子认证服务业监督管理工作的部门的工作人员，不依法履行行政许可、监督管理职责的，依法给予行政处分；构成犯罪的，依法追究刑事责任。

第五章 附则

第三十四条 本法中下列用语的含义：

（一）电子签名人，是指持有电子签名制作数据并以本人身份或者以其所代表的人的名义实施电子签名的人；

（二）电子签名依赖方，是指基于对电子签名认证证书或者电子签名的信赖从事有关活动的人；

(三) 电子签名认证证书,是指可证实电子签名人与电子签名制作数据有联系的数据电文或者其他电子记录;

(四) 电子签名制作数据,是指在电子签名过程中使用的,将电子签名与电子签名人可靠地联系起来的字符、编码等数据;

(五) 电子签名验证数据,是指用于验证电子签名的数据,包括代码、口令、算法或者公钥等。

第三十五条 国务院或者国务院规定的部门可以依据本法制定政务活动和其他社会活动中使用电子签名、数据电文的具体办法。

第三十六条 本法自 2005 年 4 月 1 日起施行。

27.2 电子认证服务管理办法

(中华人民共和国工业和信息化部令第 1 号, 2009 年 2 月 28 日发布, 自 2009 年 3 月 31 日起施行)

(中华人民共和国信息产业部令第 35 号, 2005 年 2 月 8 日发布的《电子认证服务管理办法》同时废止)

第一章 总则

第一条 为了规范电子认证服务行为,对电子认证服务提供者实施监督管理,根据《中华人民共和国电子签名法》和其他法律、行政法规的规定,制定本办法。

第二条 本办法所称电子认证服务,是指为电子签名相关各方提供真实性、可靠性验证的活动。

本办法所称电子认证服务提供者,是指为需要第三方认证的电子签名提供认证服务的机构(以下称为“电子认证服务机构”)。

向社会公众提供服务的电子认证服务机构应当依法设立。

第三条 在中华人民共和国境内设立电子认证服务机构和为电子签名提供电子认证服务,适用本办法。

第四条 中华人民共和国工业和信息化部(以下简称“工业和信息化部”)依法对电子认证服务机构和电子认证服务实施监督管理。

第二章 电子认证服务机构

第五条 电子认证服务机构应当具备下列条件:

(一) 具有独立的企业法人资格。

(二) 具有与提供电子认证服务相适应的人员。从事电子认证服务的专业技术人员、运营管理人员、安全管理人员和客户服务人员不少于三十名,并且应当符合相应岗位技能要求。

(三) 注册资本不低于人民币三千万元。

(四) 具有固定的经营场所和满足电子认证服务要求的物理环境。

(五) 具有符合国家有关安全标准的技术和设备。

(六) 具有国家密码管理机构同意使用密码的证明文件。

(七) 法律、行政法规规定的其他条件。

第六条 申请电子认证服务许可的，应当向工业和信息化部提交下列材料：

(一) 书面申请。

(二) 人员证明。

(三) 资金证明（经依法审计的近三年的财务会计报告，新成立公司的验资报告）。

(四) 经营场所证明。

(五) 国家有关认证检测机构出具的技术、设备、物理环境符合国家有关安全标准的凭证。

(六) 国家密码管理机构同意使用密码的证明文件。

第七条 工业和信息化部对提交的申请材料进行形式审查。申请材料齐全、符合法定形式的，应当向申请人出具受理通知书。申请材料不齐全或者不符合法定形式的，应当当场或者在五日内一次告知申请人需要补正的全部内容。

第八条 工业和信息化部对决定受理的申请材料进行实质审查。需要对有关内容进行核实的，指派两名以上工作人员实地进行核查。

第九条 工业和信息化部对与申请人有关事项书面征求中华人民共和国商务部等有关部门的意见。

第十条 工业和信息化部应当自接到申请之日起四十五日内作出准予许可或者不予许可的书面决定。不予许可的，应当书面通知申请人并说明理由；准予许可的，颁发《电子认证服务许可证》，并公布下列信息：

(一) 《电子认证服务许可证》编号。

(二) 电子认证服务机构名称。

(三) 发证机关和发证日期。

电子认证服务许可相关信息发生变更的，工业和信息化部应当及时公布。

《电子认证服务许可证》的有效期为五年。

第十一条 取得电子认证服务许可的，应当持《电子认证服务许可证》到工商行政管理机关办理相关手续。

第十二条 取得认证资格的电子认证服务机构，在提供电子认证服务之前，应当通过互联网公布下列信息：

(一) 机构名称和法定代表人。

(二) 机构住所和联系办法。

(三) 《电子认证服务许可证》编号。

(四) 发证机关和发证日期。

(五) 《电子认证服务许可证》有效期的起止时间。

第十三条 电子认证服务机构在《电子认证服务许可证》的有效期内拟变更公司名称、住所、法定代表人、注册资本、类型、股东以及股东的出资方式、出资额、出资时间等事项的，在向公司登记机关申请变更登记前应当报工业和信息化部同意。

第十四条 《电子认证服务许可证》的有效期届满需要延续的，电子认证服务机构应当在许可证有效期届满三十日前向工业和信息化部申请办理延续手续，并自办结之日起五日内按照本办法第十二条的规定公布相关信息。

第三章 电子认证服务

第十五条 电子认证服务机构应当按照工业和信息化部公布的《电子认证业务规则规范》等要求，制定本机构的电子认证业务规则和相应的证书策略，在提供电子认证服务前予以公布，并向工业和信息化部备案。

电子认证业务规则和证书策略发生变更的，电子认证服务机构应当予以公布，并自公布之日起三十日内向工业和信息化部备案。

第十六条 电子认证服务机构应当按照公布的电子认证业务规则提供电子认证服务。

第十七条 电子认证服务机构应当保证提供下列服务：

- (一) 制作、签发、管理电子签名认证证书。
- (二) 确认签发的电子签名认证证书的真实性。
- (三) 提供电子签名认证证书目录信息查询服务。
- (四) 提供电子签名认证证书状态信息查询服务。

第十八条 电子认证服务机构应当履行下列义务：

- (一) 保证电子签名认证证书内容在有效期内完整、准确。
- (二) 保证电子签名依赖方能够证实或者了解电子签名认证证书所载内容及其他有关事项。

- (三) 妥善保存与电子认证服务相关的信息。

第十九条 电子认证服务机构应当建立完善的内部安全管理和内部审计制度。

第二十条 电子认证服务机构应当遵守国家的保密规定，建立完善的保密制度。

电子认证服务机构对电子签名人和电子签名依赖方的资料，负有保密的义务。

第二十一条 电子认证服务机构在受理电子签名认证证书申请前，应当向申请人告知下列事项：

- (一) 电子签名认证证书和电子签名的使用条件。
- (二) 服务收费的项目和标准。
- (三) 保存和使用证书持有人信息的权限和责任。
- (四) 电子认证服务机构的责任范围。
- (五) 证书持有人的责任范围。
- (六) 其他需要事先告知的事项。

第二十二条 电子认证服务机构受理电子签名认证申请后，应当与证书申请人签订合同，明确双方的权利义务。

第四章 电子认证服务的暂停、终止

第二十三条 电子认证服务机构在《电子认证服务许可证》的有效期限内拟终止电子认证服务的，应当在终止服务六十日前向工业和信息化部报告，并办理《电子认证服务许可证》注销手续，持工业和信息化部的相关证明文件向工商行政管理机关申请办理注销登记或者变更登记。

第二十四条 电子认证服务机构拟暂停或者终止电子认证服务的，应当在暂停或者终止电子认证服务九十日前，就业务承接及其他有关事项通知有关各方。

电子认证服务机构拟暂停或者终止电子认证服务的，应当在暂停或者终止电子认证服务六十日前向工业和信息化部报告，并与其他电子认证服务机构就业务承接进行协商，作出妥善安排。

第二十五条 电子认证服务机构拟暂停或者终止电子认证服务，未能就业务承接事项与其他电子认证服务机构达成协议的，应当申请工业和信息化部安排其他电子认证服务机构承接其业务。

第二十六条 电子认证服务机构被依法吊销电子认证服务许可的，其业务承接事项按照工业和信息化部的规定处理。

第二十七条 电子认证服务机构有根据工业和信息化部的安排承接其他机构开展的电子认证服务业务的义务。

第五章 电子签名认证证书

第二十八条 电子签名认证证书应当准确载明下列内容：

- (一) 签发电子签名认证证书的电子认证服务机构名称。
- (二) 证书持有人名称。
- (三) 证书序列号。
- (四) 证书有效期。
- (五) 证书持有人的电子签名验证数据。
- (六) 电子认证服务机构的电子签名。
- (七) 工业和信息化部规定的其他内容。

第二十九条 有下列情形之一的，电子认证服务机构可以撤销其签发的电子签名认证证书：

- (一) 证书持有人申请撤销证书。
- (二) 证书持有人提供的信息不真实。
- (三) 证书持有人没有履行双方合同规定的义务。
- (四) 证书的安全性不能得到保证。
- (五) 法律、行政法规规定的其他情况。

第三十条 有下列情形之一的，电子认证服务机构应当对申请人提供的证明身份的有关材料进行查验，并对有关材料进行审查：

- (一) 申请人申请电子签名认证证书。
- (二) 证书持有人申请更新证书。
- (三) 证书持有人申请撤销证书。

第三十一条 电子认证服务机构更新或者撤销电子签名认证证书时，应当予以公告。

第六章 监督管理

第三十二条 工业和信息化部对电子认证服务机构进行定期、不定期的监督检查，监督检查的内容主要包括法律法规符合性、安全运营管理、风险管理等。

工业和信息化部对电子认证服务机构实行监督检查时，应当记录监督检查的情况和处理结果，由监督检查人员签字后归档。公众有权查阅监督检查记录。

工业和信息化部对电子认证服务机构实行监督检查，不得妨碍电子认证服务机构正常的生产经营活动，不得收取任何费用。

第三十三条 取得电子认证服务许可的电子认证服务机构，在电子认证服务许可的有效期限内不得降低其设立时所应具备的条件。

第三十四条 电子认证服务机构应当如实向工业和信息化部报送认证业务开展情况报告、财务会计报告等有关资料。

第三十五条 电子认证服务机构有下列情况之一的，应当及时向工业和信息化部报告：

（一）重大系统、关键设备事故。

（二）重大财产损失。

（三）重大法律诉讼。

（四）关键岗位人员变动。

第三十六条 电子认证服务机构应当对其从业人员进行岗位培训。

第三十七条 工业和信息化部根据监督管理工作的需要，可以委托有关省、自治区和直辖市信息产业主管部门承担具体的监督管理事项。

第七章 罚则

第三十八条 电子认证服务机构向工业和信息化部隐瞒有关情况、提供虚假材料或者拒绝提供反映其活动的真实材料的，由工业和信息化部责令改正，给予警告或者处以 5000 元以上 1 万元以下的罚款。

第三十九条 工业和信息化部与省、自治区、直辖市信息产业主管部门的工作人员，不依法履行监督管理职责的，由工业和信息化部或者省、自治区、直辖市信息产业主管部门依据职权视情节轻重，分别给予警告、记过、记大过、降级、撤职、开除的行政处分；构成犯罪的，依法追究刑事责任。

第四十条 电子认证服务机构违反本办法第十三条、第十五条、第二十七条的规定的，由工业和信息化部依据职权责令限期改正，处以警告，可以并处 1 万元以下的罚款。

第四十一条 电子认证服务机构违反本办法第三十三条的规定的，由工业和信息化部依据职权责令限期改正，处以 3 万元以下的罚款，并将上述情况向社会公告。

第八章 附则

第四十二条 经工业和信息化部根据有关协议或者对等原则核准后，中华人民共和国境外的电子认证服务机构在境外签发的电子签名认证证书与依照本办法设立电子认证服务机构签发的电子签名认证证书具有同等的法律效力。

第四十三条 本办法自 2009 年 3 月 31 日起施行。2005 年 2 月 8 日发布的《电子认证服务管理办法》（中华人民共和国信息产业部令第 35 号）同时废止。

27.3 电子认证服务密码管理办法

（国家密码管理局公告第 17 号，2009 年 10 月 28 日公布，自 2009 年 12 月 1 日起施行）

（国家密码管理局公告第 2 号，2005 年 3 月 31 日发布的《电子认证服务密码管理办法》

同时废止)

第一条 为了规范电子认证服务提供者使用密码的行为,根据《中华人民共和国电子签名法》、《商用密码管理条例》和相关法律、行政法规的规定,制定本办法。

第二条 国家密码管理局对电子认证服务提供者使用密码的行为实施监督管理。

省、自治区、直辖市密码管理机构依据本办法承担有关监督管理工作。

第三条 提供电子认证服务,应当依据本办法申请《电子认证服务使用密码许可证》。

第四条 采用密码技术为社会公众提供第三方电子认证服务的系统(以下称电子认证服务系统)使用商用密码。

电子认证服务系统应当由具有商用密码产品生产资质的单位承建。

第五条 电子认证服务系统的建设和运行应当符合《证书认证系统密码及其相关安全技术规范》。

第六条 电子认证服务系统所需密钥服务由国家密码管理局和省、自治区、直辖市密码管理机构规划的密钥管理系统提供。

第七条 申请《电子认证服务使用密码许可证》应当在电子认证服务系统建设完成后,向所在地的省、自治区、直辖市密码管理机构提交下列材料:

(一)《电子认证服务使用密码许可证申请表》;

(二)企业法人营业执照或者企业名称预先核准通知书的复印件;

(三)电子认证服务系统安全性审查相关技术材料,包括建设工作总结报告、技术工作总结报告、安全性设计报告、安全管理策略和规范报告、用户手册和测试说明;

(四)电子认证服务系统互联互通测试相关技术材料;

(五)电子认证服务系统物理环境符合电磁屏蔽、消防安全有关要求的证明文件;

(六)电子认证服务系统使用的信息安全产品符合有关法律规定的证明文件。

第八条 申请人提交的申请材料齐全并且符合规定形式的,省、自治区、直辖市密码管理机构应当受理并发给《受理通知书》;申请材料不齐全或者不符合规定形式的,省、自治区、直辖市密码管理机构应当当场或者在 5 个工作日内一次告知需要补正的全部内容。不予受理的,应当书面通知并说明理由。

省、自治区、直辖市密码管理机构应当自受理申请之日起 5 个工作日内将全部申请材料报送国家密码管理局。

第九条 国家密码管理局对省、自治区、直辖市密码管理机构报送的材料进行核查,组织对电子认证服务系统进行安全性审查和互联互通测试,并自省、自治区、直辖市密码管理机构受理申请之日起 15 个工作日内,将安全性审查和互联互通测试所需时间书面通知申请人。

电子认证服务系统通过安全性审查和互联互通测试的,由国家密码管理局发给《电子认证服务使用密码许可证》并予以公布;未通过安全性审查或者互联互通测试的,不予许可,书面通知申请人并说明理由。

第十条 《电子认证服务使用密码许可证》载明下列内容:

(一)许可证编号;

(二)电子认证服务提供者名称;

(三)许可证有效期限;

(四) 发证机关和发证日期。

《电子认证服务使用密码许可证》有效期为 5 年。

第十一条 电子认证服务提供者变更名称的,应当自变更之日起 30 日内,持变更证明文件到所在地的省、自治区、直辖市密码管理机构办理《电子认证服务使用密码许可证》更换手续。

电子认证服务提供者变更住所、法定代表人的,应当自变更之日起 30 日内,持变更证明文件到所在地的省、自治区、直辖市密码管理机构备案。

第十二条 《电子认证服务使用密码许可证》有效期满需要延续的,应当在许可证有效期届满 30 日前向国家密码管理局提出申请。国家密码管理局根据申请,在许可证有效期届满前作出是否准予延续的决定。

第十三条 电子认证服务提供者取得《电子认证服务使用密码许可证》后 6 个月内,未取得国务院信息产业主管部门颁发的《电子认证服务许可证》的,《电子认证服务使用密码许可证》自行失效。

第十四条 电子认证服务提供者终止电子认证服务或者《电子认证服务许可证》被吊销的,原持有的《电子认证服务使用密码许可证》自行失效。

第十五条 电子认证服务提供者对其电子认证服务系统进行技术改造或者进行系统搬迁的,应当将有关情况书面报国家密码管理局,经国家密码管理局同意后方可继续运行。必要时,国家密码管理局可以组织对电子认证服务系统进行安全性审查和互联互通测试。

第十六条 国家密码管理局和省、自治区、直辖市密码管理机构对电子认证服务提供者使用密码的情况进行监督检查。监督检查采取书面审查和现场核查相结合的方式。

监督检查发现存在不符合许可条件的情形的,限期整改;限期整改后仍不符合许可条件的,由国家密码管理局撤销其《电子认证服务使用密码许可证》,通报国务院信息产业主管部门并予以公布。

第十七条 有下列情形之一的,由国家密码管理局责令改正;情节严重的,吊销《电子认证服务使用密码许可证》,通报国务院信息产业主管部门并予以公布:

(一) 电子认证服务系统的运行不符合《证书认证系统密码及其相关安全技术规范》的;
(二) 电子认证服务系统使用本办法第六条规定以外的密钥管理系统提供的密钥开展业务的;

(三) 对电子认证服务系统进行技术改造或者进行系统搬迁,未按照本办法第十五条规定办理的。

第十八条 国家密码管理局和省、自治区、直辖市密码管理机构的工作人员在电子认证服务密码管理工作中滥用职权、玩忽职守、徇私舞弊的,依法给予行政处分;构成犯罪的,依法追究刑事责任。

第十九条 《电子认证服务使用密码许可证申请表》由国家密码管理局统一印制。

第二十条 本办法施行前已经取得《电子认证服务使用密码许可证》的电子认证服务提供者,应当自本办法施行之日起 3 个月内到所在地的省、自治区、直辖市密码管理机构办理《电子认证服务使用密码许可证》的换证手续。

第二十一条 本办法自 2009 年 12 月 1 日起施行。2005 年 3 月 31 日国家密码管理局发布的《电子认证服务密码管理办法》同时废止。

27.4 电子政务电子认证服务管理办法

《电子政务电子认证服务管理办法（试行）》

第一章 总则

第一条 为了规范电子政务电子认证服务活动，依据（中华人民共和国电子签名法）、（商用密码管理条例）和国务院有关文件，制定本办法。

第二条 本办法所称电子政务电子认证服务（以下简称认证服务），是指电子认证服务机构采用密码技术，通过数字证书，为各级政务部门开展社会管理、公共服务等政务活动提供的电子认证服务。

电子政务电子认证服务包括面向政务部门的电子政务电子认证服务和面向企事业单位、社会团体、社会公众的电子政务电子认证服务。

第三条 电子政务电子认证服务活动的电子认证基础设施、电子认证服务机构、电子认证服务应用等的管理，适用本办法。

第四条 国家密码管理局负责全国电子政务电子认证服务活动的监督管理工作。

各省、自治区、直辖市和中央国家机关有关部委密码管理部门（以下简称省部密码管理部门）按照国家密码管理局的统一要求，负责本地区本部门电子政务电子认证服务活动的监督管理工作。

第二章 基础设施

第五条 电子政务电子认证基础设施基于密码技术，由实现数字证书签发、注册审核和查询服务等功能的软硬件产品和系统组成。认证服务活动应当依托国家密码管理局批准的电子政务电子认证基础设施开展。

第六条 用于认证服务活动的电子政务电子认证基础设施应当具备以下条件：

- （一）符合电子政务电子认证体系建设总体规划布局要求；
- （二）采用国家密码管理局认定的商用密码产品；
- （三）由具有商用密码产品生产资质的单位承建；
- （四）符合（证书认证系统密码及其相关安全技术规范）等标准规范要求；
- （五）采用国家密码管理局规划建设的密钥管理基础设施提供密钥管理服务；
- （六）支持电子政务电子认证源点的管理；
- （七）通过国家密码管理局安全性审查。

第七条 省、自治区、直辖市密码管理部门确定的本地区电子政务电子认证基础设施，应当报经国家密码管理局批准。

第八条 电子认证服务机构跨区域建设注册审核系统，应当经所服务的省、自治区、直辖市密码管理部门批准。

第九条 中央和国家机关有关部委原则上不再建设延伸到省、自治区、直辖市的电子认证基础设施。确有特殊需要的，在提供已建电子政务电子认证基础设施不能满足其电子认证需要的相关证明后，经国家密码管理局批准可以建设相应的电子政务电子认证基础设施。

第十条 电子政务电子认证基础设施运行过程中，如有不符合本办法第六条规定的情形，应当及时整改，并报国家密码管理局重新审查。

第十一条 电子政务电子认证基础设施进行技术改造，可能对其功能或者安全性造成影响的，改造前报所属省部密码管理部门备案，改造完成后报国家密码管理局审查通过，方可投入运行。

第十二条 电子认证服务机构应当建立电子政务电子认证基础设施运行管理制度、安全访问策略、软件控制流程、内部审计机制，以及完善的灾难恢复和应急响应机制，保障安全运行。

第十三条 电子认证服务机构按年度对电子政务电子认证基础设施进行安全性评估，并对发现的问题进行整改。涉及技术改造的，按照本办法第十一条规定执行。安全性评估和整改情况报所属省部密码管理部门备案。

第三章 服务机构

第十四条 电子认证服务机构应当承担以下责任：

（一）制定本机构的电子政务电子认证服务业务规则，包括责任范围、作业操作规范、安全保障措施等事项，并在服务范围内予以发布；

（二）建立相应的服务队伍，制定相应的服务规范，提供安全可信的认证服务；

（三）保障电子政务电子认证基础设施的安全可靠运行；

（四）加强对从业人员的安全保密、职业道德和岗位技能培训。

第十五条 电子认证服务机构应当具备以下条件：

（一）事业法人或者取得电子认证服务许可的国有控股企业法人；

（二）具有经国家密码管理局批准的电子政务电子认证基础设施；

（三）具有与从事认证服务相适应的专业技术人员、运行维护人员、安全管理人员和服务人员；

（四）具有与认证服务相适应的运行服务、应用支持和安全保障等机制；

（五）法律法规规定的其他条件。

第十六条 面向企事业单位、社会团体、社会公众提供服务的电子认证服务机构，应当取得国务院信息产业主管部门颁发的《电子认证服务许可证》。

第十七条 国家密码管理局组织开展认证服务能力评估，发布《电子政务电子认证服务机构目录》。

第十八条 电子认证服务机构申请进行认证服务能力评估时，应当向所属省部密码管理部门提交以下材料：

（一）书面申请；

（二）事业单位应当提交组织机构代码证书、法人证书以及其他相关证明文件的复印件；企业应当提交组织机构代码证书、营业执照、（电子认证服务许可证）、公司章程以及其他相关证明文件的复印件；

（三）电子政务电子认证基础设施经国家密码管理局批准的证明材料的复印件；

（四）专业技术人员、运行维护人员、安全管理人员和服务人员的资格证明材料；

（五）本机构的电子认证服务业务规则；

(六) 运行服务、应用支持和安全保障等机制的相关材料；

(七) 国家密码管理局规定的其他材料。

第十九条 省部密码管理部门应当对电子认证服务机构提交的材料进行审查，并在收到材料之日起十个工作日内向申请人告知审查结果。审查通过的，应当在审查通过后五个工作日内，将全部材料报国家密码管理局。

第二十条 国家密码管理局应当自收到省部密码管理部门报送的材料之日起三十个工作日内，组织对电子认证服务机构进行能力评估。通过评估的，由国家密码管理局出具通过能力评估的证明文件，并将电子认证服务机构列入（电子政务电子认证服务机构目录），未通过评估的，书面通知申请人并说明理由。

第二十一条 电子认证服务机构应当提供以下服务内容：

(一) 数字证书的申请、签发、更新、延期、恢复、撤销等证书生命周期管理服务；

(二) 数字证书信息查询服务及数字证书状态信息查询服务；

(三) 为数字证书用户提供使用支持服务；

(四) 为政务部门提供数字证书统计、查询、下载等支持服务，以及与电子政务系统的数字证书应用集成支持服务；

(五) 提供数字证书相关培训。

第二十二条 电子认证服务机构应当按照本机构发布的电子认证服务业务规则开展认证服务。

第二十三条 电子认证服务机构跨区域提供服务的，应当接受所服务区域的省、自治区、直辖市密码管理部门的监督管理。电子认证服务机构应当将拟开展服务的范围和对象等情况，提前十个工作日书面告知所服务区域的省、自治区、直辖市密码管理部门；在开展认证服务后二十个工作日内，应当向所服务区域的省、自治区、直辖市密码管理部门备案。办理备案时，提交以下材料：

(一) 本机构开展认证服务活动的情况，包括服务范围、服务对象、服务时间等；

(二) 国家密码管理局出具的通过能力评估的证明文件的复印件；

(三) 本机构的电子认证服务业务规则，以及运行服务、应用支持和安全保障机制的相关材料。

第二十四条 电子认证服务机构应当采用符合国家相关法律法规要求的载体，完整记录、妥善保存与认证相关的信息，并承担保密责任。面向企事业单位、社会团体、社会公众的电子政务电子认证服务，信息保存期为证书失效后五年；面向政务部门的电子政务电子认证服务，信息保存期为证书失效后十年。

第二十五条 电子认证服务机构应当建立信息安全管理机制，制定相关资产、人员、物理环境等的安全策略，落实安全管理岗位和职责，并对安全策略的执行情况进行监督检查。

第二十六条 电子认证服务机构应当建立业务连续性计划，并定期开展业务风险评估，依据评估结果对本机构的电子政务电子认证服务业务规则及时进行修订，并在服务范围内予以发布。业务规则的修订，不应当降低电子认证服务机构的服务能力和水平。

第四章 服务应用

第二十七条 电子政务信息系统应当根据业务需要采用电子认证服务。

第二十八条 政务部门应当从《电子政务外网电子认证服务机构目录》中选择电子认证服务机构提供服务。

第二十九条 电子政务信息系统建设应用过程中采用电子认证服务,应当遵循国家密码管理局制定的相关标准规范。

第三十条 申请使用电子政务数字证书的机构和个人,应当向电子认证服务机构提供合法、有效的证明材料。

第三十一条 数字证书所有人应当妥善保管数字证书及其密钥。数字证书载体丢失或密钥失控时,数字证书所有人应当立即向电子认证服务机构报告,由电子认证服务机构撤销其数字证书。

第五章 监督管理

第三十二条 电子认证服务机构应当在每年的1月31日前,向国家密码管理局提交上一年度的电子政务电子认证基础设施运行管理和认证服务情况报告。

第三十三条 国家密码管理局按年度对电子认证服务机构的服务能力进行评估,不定期对电子认证服务机构的相关情况进行现场检查。

第三十四条 电子认证服务机构应当根据年度评估情况和现场检查情况,在规定期限内对存在的问题进行整改,整改期限内不得开展新的电子政务电子认证服务业务。

第三十五条 电子认证服务机构有以下情形之一的,国家密码管理局责令其停止服务,并将该机构从《电子政务外网电子认证服务机构目录》中撤销;

- (一) 未依照本办法开展认证服务活动并造成恶劣影响的;
- (二) 未通过年度评估的;
- (三) 未在规定期限内完成整改的。

第三十六条 电子政务服务机构被国家密码管理局停止电子政务电子认证服务的,其业务承接事项的处理按照国家密码管理局的要求进行。

第三十七条 电子认证服务机构拟变更机构名称、住所、法定代表人、企业股本结构或者事业单位隶属关系,以及可能对电子政务电子认证服务产生较大影响事项的,在变更前应当将拟变更事项书面告知所属省部密码管理部门。其中,变更事项影响电子认证服务机构设立条件的,应当重新进行能力评估。

第三十八条 电子认证服务机构拟暂停或者终止认证服务的,应当在暂停或者终止认证服务六十个工作日前,选定业务承接电子认证服务机构,就业务承接有关事项作出妥善安排,并在暂停或者终止认证服务四十五个工作日前向国家密码管理局报告。

第三十九条 电子认证服务机构拟暂停或者终止认证服务,不能就业务承接事项作出妥善安排的,应当在暂停或者终止认证服务六十个工作日前,向国家密码管理局提出安排其他电子认证服务机构承接业务的申请。

第四十条 电子认证服务机构有义务按照国家密码管理局的安排,承接其他电子认证服务机构的电子政务电子认证服务业务。

第四十一条 使用电子政务电子认证服务的政务部门,应当接受所属省部密码管理部门对其电子政务电子认证服务的使用情况的监督和检查。

第四十二条 未按照本办法采用电子政务电子认证服务的,应当按照国家密码管理局

或省部密码管理部门的要求限期整改。造成重大不良影响或拒不整改的，国家密码管理局或省部密码管理部门予以通报。

第四十三条 电子政务电子认证服务管理部门的相关工作人员，玩忽职守、滥用职权、徇私舞弊的，依法给与行政处分。

第六章 附则

第四十四条 面向各级政务部门内部办公、管理、协调、监督和决策的电子政务电子认证管理办法另行制定。

第四十五条 本办法自2009年11月1日起施行。

27.5 卫生系统电子认证服务管理办法

《卫生系统电子认证服务管理办法（试行）》

第一章 总则

第一条 为保障卫生信息系统安全，规范卫生系统电子认证服务体系建设，依据《中华人民共和国电子签名法》和《电子认证服务管理办法》（工业和信息化部令2009年第1号），结合卫生系统业务特点，制定本办法。

第二条 本办法适用于在全国卫生系统范围内管理、使用和提供电子认证服务的管理方、使用方和提供方。管理方包括卫生部和各省级卫生行政部门信息化工作领导小组；使用方包括全国各级卫生行政部门和各级各类医疗卫生机构及其工作人员、涉及卫生业务的企事业单位和社会公众；提供方包括为卫生系统提供电子认证服务的电子认证服务机构。

第三条 本办法所称电子认证服务，是指电子认证服务机构按照卫生系统电子认证服务体系相关技术标准和服务规范，为使用方提供数字证书的颁发、更新、吊销、密码解锁、密钥恢复以及证书应用等服务，从而满足卫生信息系统在身份认证、授权管理、责任认定等方面的信息安全需求。

第四条 坚持合法性原则、应用导向原则、市场竞争原则、一证多用原则，依托依法设立的第三方电子认证服务机构，按照“政府监管、企业承办”的方式，建设卫生系统电子认证服务体系。

第五条 卫生部信息化工作领导小组负责全国卫生系统电子认证服务体系的建设和管理工作，负责组织制定卫生系统电子认证服务相关技术标准和服务规范。各省级卫生行政部门信息化工作领导小组负责指导、推进本辖区卫生信息系统开展安全、规范的电子认证服务应用。

第二章 电子认证服务机构的管理

第六条 电子认证服务机构面向卫生系统提供电子认证服务应当具备以下必要条件：

（一）取得工业和信息化部颁发的《电子认证服务许可证》；

（二）符合《电子政务电子认证体系建设总体规划》（国密局联字〔2007〕2号）中关于电子认证体系建设的相关要求；

(三) 具有符合《卫生系统电子认证服务规范》、《卫生系统数字证书格式规范》、《卫生系统数字证书介质技术规范》、《卫生系统数字证书应用集成规范》和《卫生系统数字证书服务管理平台接入规范》等的电子认证服务体系;

(四) 符合法律、行政法规规定的其他条件。

第七条 卫生部信息化工作领导小组办公室负责选择卫生部及其直属单位的电子认证服务机构。各省级卫生行政部门信息化工作领导小组办公室负责选择本辖区的电子认证服务机构。

第八条 各省级卫生行政部门信息化工作领导小组办公室选定电子认证服务机构后,应当将服务机构名单及其相关材料向卫生部信息化工作领导小组办公室上报,并要求电子认证服务机构在开展服务前接入卫生部数字证书服务管理平台。

第九条 鼓励各地卫生系统数字证书应用单位优先选择已接入卫生部数字证书服务管理平台的电子认证服务机构提供服务。

第十条 卫生部信息化工作领导小组办公室与各省级卫生行政部门信息化工作领导小组办公室共同对各电子认证服务机构的服务质量及开展情况进行定期检查。

第十一条 卫生部信息化工作领导小组办公室通过数字证书服务管理平台收集、汇总电子认证服务机构面向卫生系统证书发放和服务质量反馈等信息,综合掌握全国卫生系统电子认证服务的整体应用情况。

第三章 电子认证服务机构的服务要求

第十二条 电子认证服务机构应当按照《卫生系统电子认证服务规范》的要求,建立全面、规范、安全、高效的运行服务体系,并至少提供以下服务:

- (一) 数字证书的颁发、更新、吊销、密码解锁、密钥恢复等服务;
- (二) 数字证书及黑名单的查询与下载服务;
- (三) 为数字证书用户提供应用支持服务;
- (四) 提供数字证书相关的培训服务。

第十三条 电子认证服务机构应当妥善保存数字证书业务申请材料,并承担保密责任,不得泄露或遗失,信息保存期至少为证书到期后 5 年。

第十四条 电子认证服务机构应当建立信息安全保障机制,针对相关资产、人员、物理环境和软件系统等制定安全策略及管理制度,采取有效的安全保障措施,并对安全策略的执行情况进行有效的监督检查,确保运行服务安全可靠,满足卫生信息系统业务连续性要求。

第四章 数字证书的应用管理

第十五条 凡涉及国家安全、社会稳定、公众利益等方面的各类重要卫生信息系统,应当按照国家法律法规、信息安全等级保护制度等要求,采用电子认证服务,解决身份认证、授权管理、责任认定等安全问题,主要包括:

- (一) 涉及公共卫生业务的信息系统;
- (二) 涉及医疗保健的医疗卫生信息系统;
- (三) 网上申报、年检、备案、资质认定等行政审批信息系统;
- (四) 各类网上招标采购信息系统;

(五) 其他重要卫生信息系统。

第十六条 使用电子认证服务的各类卫生信息系统, 应当遵循《卫生系统数字证书应用集成规范》进行建设, 实现数字证书的各项安全功能。

第十七条 纳入卫生系统电子认证服务体系的电子认证服务机构, 其颁发的数字证书应当能够在各类卫生信息系统中进行注册、授权及使用, 确保实现互信互认、一证多用。

第十八条 数字证书使用单位应当加强证书管理, 指导并要求证书持有人妥善保管数字证书介质。出现数字证书介质丢失或损坏等异常情况时, 证书持有人应当立即联系电子认证服务机构进行妥善处理。

第十九条 数字证书原则上由信息系统建设单位统一采购配发并负责初次办理费用, 其年服务费用由使用单位负责; 对于向社会提供服务的信息系统, 其费用由服务申请者支付。

第二十条 多个卫生信息系统覆盖相同用户群体时, 由卫生部和各省级卫生行政部门信息化工作领导小组办公室协调各信息系统建设单位, 分摊数字证书的初次办理费用。

第五章 附则

第二十一条 各省级卫生行政部门信息化工作领导小组应当按照本办法第六条要求, 对本辖区已开展服务的电子认证服务机构进行评估。符合第六条各项条件的, 方可接入卫生部数字证书服务管理平台并继续开展服务; 不符合第六条规定条件的, 应当责令其进行整改, 如 1 年内仍达不到相关要求的, 应当终止其服务。

第二十二条 已建成但尚未采用数字证书的重要卫生信息系统, 应当尽快采用数字证书, 实现身份认证、授权管理和责任认定; 已经采用数字证书的重要卫生信息系统应当尽快按照本办法的有关要求进行系统改造, 纳入卫生系统电子认证服务体系。

第二十三条 本办法由卫生部信息化工作领导小组办公室负责解释。

第二十四条 本办法自 2010 年 1 月 1 日起试行。

27.6 商用密码管理条例

(中华人民共和国国务院第 273 号令, 1999 年 10 月 7 日发布, 自发布之日起施行)

第一章 总则

第一条 为了加强商用密码管理, 保护信息安全, 保护公民和组织的合法权益, 维护国家的安全和利益, 制定本条例。

第二条 本条例所称商用密码, 是指对不涉及国家秘密内容的信息进行加密保护或者安全认证所使用的密码技术和密码产品。

第三条 商用密码技术属于国家秘密。国家对商用密码产品的科研、生产、销售和使用实行专控管理。

第四条 国家密码管理委员会及其办公室(以下简称国家密码管理机构)主管全国的商用密码管理工作。

省、自治区、直辖市负责密码管理的机构根据国家密码管理机构的委托, 承担商用密

码的有关管理工作。

第二章 科研、生产管理

第五条 商用密码的科研任务由国家密码管理机构指定的单位承担。

商用密码指定科研单位必须具有相应的技术力量和设备，能够采用先进的编码理论和技术，编制的商用密码算法具有较高的保密强度和抗攻击能力。

第六条 商用密码的科研成果，由国家密码管理机构组织专家按照商用密码技术标准和技术规范审查、鉴定。

第七条 商用密码产品由国家密码管理机构指定的单位生产。未经指定，任何单位或者个人不得生产商用密码产品。

商用密码产品指定生产单位必须具有与生产商用密码产品相适应的技术力量以及确保商用密码产品质量的设备、生产工艺和质量保证体系。

第八条 商用密码产品指定生产单位生产的商用密码产品的品种和型号，必须经国家密码管理机构批准，并不得超过批准范围生产商用密码产品。

第九条 商用密码产品，必须经国家密码管理机构指定的产品质量检测机构检测合格。

第三章 销售管理

第十条 商用密码产品由国家密码管理机构许可的单位销售。未经许可，任何单位或者个人不得销售商用密码产品。

第十一条 销售商用密码产品，应当向国家密码管理机构提出申请，并应当具备下列条件：

- (一) 有熟悉商用密码产品知识和承担售后服务的人员；
- (二) 有完善的销售服务和安全管理规章制度；
- (三) 有独立的法人资格。

经审查合格的单位，由国家密码管理机构发给《商用密码产品销售许可证》。

第十二条 销售商用密码产品，必须如实登记直接使用商用密码产品的用户的名称(姓名)、地址(住址)、组织机构代码(居民身份证号码)以及每台商用密码产品的用途，并将登记情况报国家密码管理机构备案。

第十三条 进口密码产品以及含有密码技术的设备或者出口商用密码产品，必须报经国家密码管理机构批准。任何单位或者个人不得销售境外的密码产品。

第四章 使用管理

第十四条 任何单位或者个人只能使用经国家密码管理机构认可的商用密码产品，不得使用自行研制的或者境外生产的密码产品。

第十五条 境外组织或者个人在中国境内使用密码产品或者含有密码技术的设备，必须报经国家密码管理机构批准；但是，外国驻华外交代表机构、领事机构除外。

第十六条 商用密码产品的用户不得转让其使用的商用密码产品。商用密码产品发生故障，必须由国家密码管理机构指定的单位维修。报废、销毁商用密码产品，应当向国家密码管理机构备案。

第五章 安全、保密管理

第十七条 商用密码产品的科研、生产，应当在符合安全、保密要求的环境中进行。销售、运输、保管商用密码产品，应当采取相应的安全措施。

从事商用密码产品的科研、生产和销售以及使用商用密码产品的单位和人员，必须对所接触和掌握的商用密码技术承担保密义务。

第十八条 宣传、公开展览商用密码产品，必须事先报国家密码管理机构批准。

第十九条 任何单位和个人不得非法攻击商用密码，不得利用商用密码危害国家的安全和利益、危害社会治安或者进行其他违法犯罪活动。

第六章 罚则

第二十条 有下列行为之一的，由国家密码管理机构根据不同情况分别会同工商行政管理、海关等部门没收密码产品，有违法所得的，没收违法所得；情节严重的，可以并处违法所得1至3倍的罚款：

（一）未经指定，擅自生产商用密码产品的，或者商用密码产品指定生产单位超过批准范围生产商用密码产品的；

（二）未经许可，擅自销售商用密码产品的；

（三）未经批准，擅自进口密码产品以及含有密码技术的设备、出口商用密码产品或者销售境外的密码产品的。

经许可销售商用密码产品的单位未按照规定销售商用密码产品的，由国家密码管理机构会同工商行政管理部门给予警告，责令改正。

第二十一条 有下列行为之一的，由国家密码管理机构根据不同情况分别会同公安、国家安全机关给予警告，责令立即改正：

（一）商用密码产品的科研、生产过程中违反安全、保密规定的；

（二）销售、运输、保管商用密码产品，未采取相应的安全措施的；

（三）未经批准，宣传、公开展览商用密码产品的；

（四）擅自转让商用密码产品或者不到国家密码管理机构指定的单位维修商用密码产品的。

使用自行研制的或者境外生产的密码产品，转让商用密码产品，或者不到国家密码管理机构指定的单位维修商用密码产品，情节严重的，由国家密码管理机构根据不同情况分别会同公安、国家安全机关没收其密码产品。

第二十二条 商用密码产品的科研、生产、销售单位有本条例第二十条、第二十一条第一款第（一）、（二）、（三）项所列行为，造成严重后果的，由国家密码管理机构撤销其指定科研、生产单位资格，吊销《商用密码产品销售许可证》。

第二十三条 泄露商用密码技术秘密、非法攻击商用密码或者利用商用密码从事危害国家的安全和利益的活动，情节严重，构成犯罪的，依法追究刑事责任。

有前款所列行为尚不构成犯罪的，由国家密码管理机构根据不同情况分别会同国家安全机关或者保密部门没收其使用的商用密码产品，对有危害国家安全行为的，由国家安全机关依法处以行政拘留；属于国家工作人员的，并依法给予行政处分。

第二十四条 境外组织或者个人未经批准，擅自使用密码产品或者含有密码技术的设备的，由国家密码管理机构会同公安机关给予警告，责令改正，可以并处没收密码产品或者含有密码技术的设备。

第二十五条 商用密码管理机构的工作人员滥用职权、玩忽职守、徇私舞弊，构成犯罪的，依法追究刑事责任；尚不构成犯罪的，依法给予行政处分。

第七章 附则

第二十六条 国家密码管理委员会可以依据本条例制定有关的管理规定。

第二十七条 本条例自发布之日起施行。

27.7 商用密码科研管理规定

（国家密码管理局公告第4号，2005年12月11日公布，自2006年1月1日起施行）

第一条 为了加强商用密码科研管理，促进商用密码技术进步，根据《商用密码管理条例》，制定本规定。

第二条 商用密码体制、协议、算法及其技术规范的科研活动适用本规定，学术和理论研究除外。

第三条 国家密码管理局主管全国的商用密码科研管理工作。

第四条 商用密码科研由国家密码管理局指定的单位（以下称商用密码科研定点单位）承担。

第五条 国家密码管理局根据商用密码发展以及科研的需要，指定商用密码科研定点单位。

商用密码科研定点单位必须具备独立法人资格，具有从事密码科研相应的技术力量和设备，能够采用先进的编码理论和技术，编制具有较高保密强度和抗攻击能力的商用密码算法。

第六条 国家密码管理局指定商用密码科研定点单位原则上定期集中进行。

指定商用密码科研定点单位的程序如下：

- （一）申请单位填写《商用密码科研定点单位申请表》，提交国家密码管理局；
- （二）对申请单位提交的书面材料进行初审，提出初审意见；
- （三）对通过初审的单位进行实地考察，必要时组织专家进行评估；
- （四）作出指定决定并告知指定结果。

第七条 被指定为商用密码科研定点单位的，由国家密码管理局发给《商用密码科研定点单位证书》并予以公布。

《商用密码科研定点单位证书》有效期6年。

国家密码管理局对商用密码科研定点单位每年考核一次。考核不合格的，撤销其商用密码科研定点单位资质。

第八条 商用密码科研定点单位变更名称的，应当自变更之日起30日内，持变更证明文件到国家密码管理局更换《商用密码科研定点单位证书》。

商用密码科研定点单位变更住所、法定代表人的，应当自变更之日起30日内，持变更

证明文件到国家密码管理局备案。

商用密码科研定点单位变更隶属关系、资本结构或者商用密码科研能力发生重大变化的，应当报告国家密码管理局。不适宜继续从事商用密码科研的，由国家密码管理局撤销其科研定点单位资质。

第九条 商用密码科研项目由国家密码管理局下达或者商用密码科研定点单位自选。

第十条 国家密码管理局采用任务书的形式下达项目。任务书应当明确项目名称、设计要求、技术指标、进度要求、成果形式等。

国家密码管理局依据任务书对项目的进展情况进行检查。

第十一条 商用密码科研定点单位研究完成下达的项目后，应当向国家密码管理局申请验收。

申请项目验收，应当提交下列材料：

- (一) 验收申请；
- (二) 研究工作总结报告；
- (三) 编制方案及说明；
- (四) 安全性分析报告。

申请验收商用密码算法项目的，还应当提交商用密码算法源程序、程序说明和算法工程实现的评估报告。

第十二条 国家密码管理局组织专家对申请验收的项目进行评审，并根据专家评审意见作出是否同意通过验收的决定。同意通过验收的，国家密码管理局发给证明文件。

第十三条 通过验收的商用密码科研项目成果，由国家密码管理局组织推广应用。

未通过国家密码管理局验收的商用密码科研项目成果不得投入应用。

第十四条 商用密码科研定点单位自选项目应当向国家密码管理局备案，并提交项目可行性研究报告。

项目完成后，应当向国家密码管理局申请成果鉴定。成果鉴定及成果的推广应用参照本规定第十一条、第十二条、第十三条办理。

第十五条 商用密码科研定点单位及其人员，应当对所接触和掌握的商用密码技术承担保密义务。

第十六条 商用密码科研定点单位应当建立健全保密规章制度，对其人员进行保密教育。

第十七条 商用密码科研活动应当在符合安全保密要求的环境中进行。

商用密码技术资料应当由专人保管，并采取相应的保密措施，防止商用密码技术的泄露。

第十八条 商用密码科研定点单位不得雇用外籍人员或者境外人员参与商用密码科研活动。

第十九条 参与商用密码科研项目评审、管理的专家和工作人员，应当对研究内容、技术方案和科研成果承担保密义务。

第二十条 违反本规定的行为，依照《商用密码管理条例》予以处罚。

第二十一条 《商用密码科研定点单位证书》由国家密码管理局印制。

第二十二条 本规定自2006年1月1日起施行。

27.8 商用密码产品生产管理规定

(国家密码管理局公告第 5 号, 2005 年 12 月 11 日公布, 自 2006 年 1 月 1 日起施行)

第一条 为了加强商用密码产品生产管理, 规范商用密码产品生产活动, 根据《商用密码管理条例》, 制定本规定。

第二条 本规定所称商用密码产品, 是指采用密码技术对不涉及国家秘密内容的信息进行加密保护或者安全认证的产品。

第三条 商用密码产品生产活动适用本规定。本规定所称商用密码产品生产包括商用密码产品的研制开发。

第四条 商用密码产品由国家密码管理局指定的单位(以下称商用密码产品生产定点单位)生产。未经指定, 任何单位和个人不得生产商用密码产品。

商用密码产品的品种和型号必须经国家密码管理局批准。

第五条 国家密码管理局主管全国的商用密码产品生产管理工作。

省、自治区、直辖市密码管理机构依据本规定承担有关管理工作。

第六条 国家密码管理局根据商用密码发展的需要, 指定商用密码产品生产定点单位。

商用密码产品生产定点单位必须具备独立的法人资格, 具有与开发、生产商用密码产品相适应的技术力量和场所, 具有确保商用密码产品质量的设备、生产工艺和质量保证体系, 满足法律、行政法规规定的其他条件。

第七条 国家密码管理局指定商用密码产品生产定点单位原则上定期集中进行。

指定商用密码产品生产定点单位的程序如下:

(一) 申请单位填写《商用密码产品生产定点单位申请表》, 提交所在地的省、自治区、直辖市密码管理机构;

(二) 省、自治区、直辖市密码管理机构对申请单位提交的书面材料进行初审, 提出初审意见;

(三) 国家密码管理局对通过初审的单位进行实地考察;

(四) 国家密码管理局作出指定决定并告知指定结果。

第八条 被指定为商用密码产品生产定点单位的, 由国家密码管理局发给《商用密码产品生产定点单位证书》并予以公布。

《商用密码产品生产定点单位证书》有效期 3 年。

第九条 商用密码产品生产定点单位应当自取得《商用密码产品生产定点单位证书》之日起 30 日内, 到所在地的工商行政管理部门办理许可经营项目登记手续。

第十条 国家密码管理局对商用密码产品生产定点单位进行年度考核。

商用密码产品生产定点单位应当于每年 3 月 1 日之前, 填写《商用密码产品生产定点单位考核表》并交所在地的省、自治区、直辖市密码管理机构。省、自治区、直辖市密码管理机构应当于 3 月 31 日之前将考核意见报国家密码管理局。

考核结果由国家密码管理局公布。考核不合格的, 撤销其商用密码产品生产定点单位资质。商用密码产品生产定点单位未在规定期限内填报考核材料的, 视为考核不合格。

取得《商用密码产品生产定点单位证书》未满 6 个月的, 不参加该年度的考核。

第十一条 商用密码产品生产定点单位变更名称的,应当自变更之日起30日内,持变更证明文件到所在地的省、自治区、直辖市密码管理机构办理《商用密码产品生产定点单位证书》更换手续。

商用密码产品生产定点单位变更住所、法定代表人的,应当自变更之日起30日内,持变更证明文件到所在地的省、自治区、直辖市密码管理机构备案。

商用密码产品生产定点单位破产、解散或者被撤销的,原持有的《商用密码产品生产定点单位证书》自行失效。

第十二条 商用密码产品生产定点单位生产商用密码产品,应当在研制出产品样品后向国家密码管理局申请产品品种和型号。

申请产品品种和型号应当向所在地的省、自治区、直辖市密码管理机构提交下列材料:

- (一) 商用密码产品品种和型号申请书;
- (二) 技术工作总结报告;
- (三) 安全性设计报告;
- (四) 用户手册;
- (五) 测试说明。

商用密码产品所采用的密码算法应当是国家密码管理局认可的算法。

第十三条 商用密码产品生产定点单位提交的申请材料齐备并且符合规定形式的,省、自治区、直辖市密码管理机构应当受理并发给《受理通知书》;申请材料不齐备或者不符合规定形式的,省、自治区、直辖市密码管理机构应当当场或者在5个工作日内一次告知需要补正的全部内容。不予受理的,应当书面通知并说明理由。

省、自治区、直辖市密码管理机构应当自受理申请之日起5个工作日内完成初审,并将初审意见和全部申请材料报送国家密码管理局。

国家密码管理局收到省、自治区、直辖市密码管理机构报送的材料后应当组织安全性审查(含产品样品测试,下同),并自省、自治区、直辖市密码管理机构受理申请之日起20个工作日内,将安全性审查所需时间书面告知申请人。

通过安全性审查的,在5个工作日内批给产品品种和型号,发给产品品种和型号证书,并予以公布。未通过安全性审查的,不予批准并说明理由。

安全性审查所需时间不计算在本规定所设定的期限内。

第十四条 商用密码产品生产定点单位应当按照批准的品种和型号生产产品,并在产品上标明产品型号。

第十五条 商用密码产品必须经国家指定的机构检测、认证合格,并加施强制性认证标志后方可出厂。

暂未列入强制性认证目录的商用密码产品,必须经国家密码管理局指定的产品质量检测机构检测合格。

第十六条 商用密码产品生产定点单位及其人员,应当对所接触和掌握的商用密码技术承担保密义务。

第十七条 商用密码产品生产定点单位应当建立健全保密规章制度,对其人员进行保密教育。

第十八条 生产商用密码产品应当在符合安全保密要求的环境中进行。

保管商用密码产品应当采取相应的安全措施。

第十九条 商用密码技术资料、关键部件应当由专人保管，并采取相应的保密措施，防止商用密码技术的泄露。生产过程中产生的废弃品应当妥善销毁。

第二十条 参与商用密码产品安全性审查的专家和工作人员，应当对商用密码产品的技术方案和安全设计方案承担保密义务。

第二十一条 违反本规定的行为，依照《商用密码管理条例》予以处罚。

第二十二条 《商用密码产品生产定点单位证书》由国家密码管理局印制。

第二十三条 本规定自 2006 年 1 月 1 日起施行。

27.9 商用密码产品销售管理规定

（国家密码管理局公告第 6 号，2005 年 12 月 11 日公布，自 2006 年 1 月 1 日起施行）

第一条 为了加强商用密码产品销售管理，规范商用密码产品销售行为，根据《商用密码管理条例》，制定本规定。

第二条 商用密码产品销售活动适用本规定。

第三条 本规定所称商用密码产品，是指采用密码技术对不涉及国家秘密内容的信息进行加密保护或者安全认证的产品。

第四条 国家对商用密码产品销售实行许可制度。销售商用密码产品应当取得《商用密码产品销售许可证》。

未经许可，任何单位和个人不得销售商用密码产品。

第五条 国家密码管理局主管全国的商用密码产品销售管理工作。

省、自治区、直辖市密码管理机构依据本规定承担有关管理工作。

第六条 申请《商用密码产品销售许可证》的单位应当具备下列条件：

- （一）有独立的法人资格；
- （二）有熟悉商用密码产品知识和承担售后服务的人员以及相应的资金保障；
- （三）有完善的销售服务和安全保密管理制度；
- （四）法律、行政法规规定的其他条件。

第七条 申请《商用密码产品销售许可证》应当向所在地的省、自治区、直辖市密码管理机构提交下列材料：

- （一）《商用密码产品销售许可证申请表》；
- （二）企业法人营业执照、税务登记证复印件；
- （三）证明其符合本规定第六条第（二）项、第（三）项、第（四）项所列条件的材料。

第八条 申请单位提交的材料齐备并且符合规定形式的，省、自治区、直辖市密码管理机构应当受理并发给《受理通知书》；申请材料不齐备或者不符合规定形式的，省、自治区、直辖市密码管理机构应当当场或者在 5 个工作日内一次告知需要补正的全部内容。不予受理的，应当书面通知并说明理由。

省、自治区、直辖市密码管理机构应当自受理申请之日起 5 个工作日内完成初审，并将初审意见和全部申请材料报送国家密码管理局。

国家密码管理局应当自受理申请之日起 20 个工作日内作出是否批准的决定。

国家密码管理局认为必要时,可以对申请单位进行现场考察。考察所需时间不计算在本规定所设定的期限内。

第九条 国家密码管理局批准申请单位从事商用密码产品销售活动的,应当自作出批准决定之日起 10 个工作日内发给《商用密码产品销售许可证》,并向社会公布。

《商用密码产品销售许可证》有效期 3 年。

第十条 取得《商用密码产品销售许可证》的单位(以下称商用密码产品销售许可单位),应当自取得《商用密码产品销售许可证》之日起 30 日内,到所在地的工商行政管理部门办理许可经营项目登记手续。

第十一条 商用密码产品销售许可单位设立分支机构销售商用密码产品的,应当自设立之日起 30 日内,持分支机构营业执照到分支机构所在地的省、自治区、直辖市密码管理机构备案。

第十二条 商用密码产品销售许可单位变更名称的,应当自变更之日起 30 日内,持变更证明文件到所在地的省、自治区、直辖市密码管理机构办理《商用密码产品销售许可证》更换手续。

商用密码产品销售许可单位变更住所、法定代表人的,应当自变更之日起 30 日内,持变更证明文件到所在地的省、自治区、直辖市密码管理机构备案。

商用密码产品销售许可单位破产、解散或者被撤销的,原持有的《商用密码产品销售许可证》自行失效。

第十三条 商用密码产品销售许可单位销售的商用密码产品,应当是经国家指定的机构检测、认证合格并加施强制性认证标志的产品。

商用密码产品销售许可单位销售的暂未列入强制性认证目录的商用密码产品,应当是经国家密码管理局指定的产品质量检测机构检测合格的产品。

商用密码产品销售许可单位不得销售境外研制生产的密码产品。

第十四条 商用密码产品销售许可单位,应当详细登记直接使用商用密码产品的用户的名称(姓名)、住所、组织机构代码(居民身份证号码)以及产品的名称、型号、用途、数量。

商用密码产品销售许可单位应当每季度将销售登记情况如实报所在地的省、自治区、直辖市密码管理机构备案。

省、自治区、直辖市密码管理机构应当及时将商用密码产品销售汇总情况报国家密码管理局备案。

第十五条 商用密码产品销售许可单位将商用密码产品销售给在华的境外组织或者个人的,应当核验其持有的密码产品准用证。

第十六条 商用密码产品销往境外的,按照密码产品出口管理规定办理。

第十七条 宣传、公开展览商用密码产品,应当事先报所在地的省、自治区、直辖市密码管理机构批准。

第十八条 商用密码产品销售许可单位及其人员,应当对所接触和掌握的商用密码技术承担保密义务。

第十九条 销售、运输、保管商用密码产品应当采取相应的安全措施。

第二十条 商用密码产品销售许可单位应当严格执行安全保密管理制度,对其人员进

行保密教育。

第二十一条 违反本规定的行为，依照《商用密码管理条例》予以处罚。

第二十二条 《商用密码产品销售许可证》由国家密码管理局印制。

第二十三条 本规定自 2006 年 1 月 1 日起施行。

27.10 商用密码产品使用管理规定

（国家密码管理局公告第 8 号，2007 年 3 月 24 日公布，自 2007 年 5 月 1 日起施行）

第一条 为了规范商用密码产品使用行为，根据《商用密码管理条例》，制定本规定。

第二条 中国公民、法人和其他组织使用商用密码产品的行为适用本规定。

第三条 本规定所称商用密码产品，是指采用密码技术对不涉及国家秘密内容的信息进行加密保护或安全认证的产品。

第四条 国家密码管理局主管全国的商用密码产品使用管理工作。

省、自治区、直辖市密码管理机构依据本规定承担有关管理工作。

第五条 中国公民、法人和其他组织需要对不涉及国家秘密内容的信息进行加密保护或安全认证的，均可以使用商用密码产品。

使用商用密码产品，应当遵守国家法律，不得损害国家利益、社会公共利益和其他公民的合法权益，不得利用商用密码产品进行违法犯罪活动。

第六条 中国公民、法人和其他组织都应当使用国家密码管理局准予销售的商用密码产品，不得使用自行研制的或境外生产的密码产品。

国家密码管理局定期公布准予销售的商用密码产品目录。

第七条 需要使用商用密码产品的，应当到商用密码产品销售许可单位购买。

购买商用密码产品应当向商用密码产品销售许可单位出示本人身份证，说明直接使用商用密码产品的用户名称（姓名）、地址（住址）以及产品用途，提供用户组织机构代码证（居民身份证）复印件。

第八条 需要维修商用密码产品的，应当交该产品的生产单位或销售单位维修。

第九条 外商投资企业（包括中外合资经营企业、中外合作经营企业、外资企业、外商投资股份有限公司等）确因业务需要，必须使用境外生产的密码产品与境外进行互联互通的，经国家密码管理局批准，可以使用境外生产的密码产品。

外商投资企业申请使用境外生产的密码产品，应当事先填写《使用境外生产的密码产品登记表》，交所在地的省、自治区、直辖市密码管理机构。

省、自治区、直辖市密码管理机构自受理申请之日起 5 个工作日内，对《使用境外生产的密码产品登记表》进行审查并报国家密码管理局。

国家密码管理局应当自省、自治区、直辖市密码管理机构受理申请之日起 20 个工作日内，对《使用境外生产的密码产品登记表》进行审核。准予使用的，发给《使用境外生产的密码产品准用证》。

《使用境外生产的密码产品准用证》有效期 3 年。

第十条 使用境外生产的密码产品的外商投资企业的名称、地址、密码产品用途发生变更的，应当自变更之日起 10 日内，到所在地的省、自治区、直辖市密码管理机构办理《使

用境外生产的密码产品准用证》更换手续。

第十一条 外商投资企业终止使用境外生产的密码产品的，应当自终止使用之日起30日内，书面告知所在地的省、自治区、直辖市密码管理机构，并交回《使用境外生产的密码产品准用证》。

第十二条 外商投资企业申请使用的密码产品需要从境外进口的，应当申请办理《密码产品进口许可证》。

密码产品入境时，外商投资企业应当向海关如实申报并提交《密码产品进口许可证》，海关凭此办理验放手续。

第十三条 用户不得转让其使用的密码产品。

第十四条 违反本规定的行为，依照《商用密码管理条例》予以处罚。

第十五条 《使用境外生产的密码产品登记表》、《使用境外生产的密码产品准用证》、《密码产品进口许可证》由国家密码管理局统一印制。

第十六条 本规定自2007年5月1日起施行。

27.11 境外组织和个人在华使用密码产品管理办法

（国家密码管理局公告第9号，2007年3月24日公布，自2007年5月1日起施行）

第一条 为了规范境外组织和个人在中国境内使用密码产品以及含有密码技术的设备（以下统称密码产品）的行为，根据《商用密码管理条例》，制定本办法。

第二条 境外组织和个人在中国境内使用密码产品的行为适用本办法。外国驻华使馆、领事机构，国际组织驻华代表机构等享有相应特权和豁免权的机构除外。

第三条 本办法所称境外组织，是指依照外国法律在中国境外成立的组织，包括这些组织在中国境内设立的分支机构、办事机构、代表机构等。

本办法所称境外个人，是指依照《中华人民共和国国籍法》不具有中国国籍的人。

本办法所称密码产品，是指采用密码技术对信息进行加密保护或安全认证的产品，包括境外生产的密码产品和中国生产的密码产品。

第四条 国家密码管理局主管境外组织和个人在中国境内使用密码产品的管理工作。

省、自治区、直辖市密码管理机构依据本办法承担有关管理工作。

第五条 境外组织或个人在中国境内使用密码产品，应当事先填写《境外组织或个人使用密码产品申报登记表》，交所在地的省、自治区、直辖市密码管理机构。

省、自治区、直辖市密码管理机构应当自受理申请之日起5个工作日内，对《境外组织或个人使用密码产品申报登记表》进行审查并报国家密码管理局。

国家密码管理局应当自省、自治区、直辖市密码管理机构受理申请之日起20个工作日内，对《境外组织或个人使用密码产品申报登记表》进行审核。准予使用的，发给《境外组织或个人使用密码产品准用证》。

《境外组织或个人使用密码产品准用证》有效期3年。

第六条 境外组织或个人使用的密码产品需要从境外进口的，应当申请办理《密码产品进口许可证》。

密码产品入境时，境外组织或个人应当向海关如实申报并提交《密码产品进口许可证》，

海关凭此办理验放手续。

第七条 境外组织或个人使用中国生产的密码产品，应当到中国商用密码产品销售许可单位购买，并出示《境外组织或个人使用密码产品准用证》。

第八条 使用密码产品的境外组织或个人的名称（姓名）、地址（住址）、密码产品用途发生变更的，应当自变更之日起 10 日内，到所在地的省、自治区、直辖市密码管理机构办理《境外组织或个人使用密码产品准用证》更换手续。

第九条 境外组织或个人终止使用密码产品的，应当自终止使用之日起 30 日内，书面告知所在地的省、自治区、直辖市密码管理机构，并交回《境外组织或个人使用密码产品准用证》。

第十条 境外组织和个人不得转让其使用的密码产品。

第十一条 境外组织和个人在中国境内使用密码产品，应当遵守中国法律，不得危害中国国家安全、损害社会公共利益、破坏社会公共秩序。

第十二条 违反本办法的行为，依照《商用密码管理条例》予以处罚。

第十三条 《境外组织或个人使用密码产品申报登记表》、《境外组织或个人使用密码产品准用证》、《密码产品进口许可证》由国家密码管理局统一印制。

第十四条 香港特别行政区、澳门特别行政区、台湾地区的组织和个人在内地使用密码产品的行为，参照本办法进行管理。

第十五条 本办法自 2007 年 5 月 1 日起施行。

第 28 章 国内标准

28.1 通用性标准

28.1.1 祖冲之序列密码算法（GM/T 0001）

GM/T 0001-2012《祖冲之序列密码算法》

本规范包含 3 个部分：算法描述、基于祖冲之算法的机密性算法、基于祖冲之算法的完整性算法。

第 1 部分为算法描述，描述了祖冲之序列密码算法，可用于指导祖冲之算法相关产品的研制、检测和使用。本部分主要包括算法整体结构、线性反馈移位寄存器 LFSR、比特重组 BR、非线性函数 F、密钥装入和算法运行内容，并在附录中给出了算法计算实例。

第 2 部分为基于祖冲之算法的机密性算法，可适用于 3GPP LTE 通信中的加密和解密，可用于指导基于祖冲之算法的机密性算法的相关产品的研制、检测和使用。本部分主要包括算法输入与输出和算法工作流程内容，并在附录中给出了算法计算实例。

第 3 部分为基于祖冲之算法的完整性算法，可适用于 3GPP LTE 通信中消息的完整性保护，可用于指导基于祖冲之算法的完整性算法的相关产品的研制、检测和使用。本部分主要包括算法输入与输出和算法工作流程内容，并在附录中给出了算法计算实例。

28.1.2 SM4 分组密码算法（GM/T 0002）

GM/T 0002-2012《SM4 分组密码算法》

本规范规定了 SM4 分组密码算法（原 SMS4）的算法结构和算法描述，并给出了运算示例，适用于密码应用中使用分组密码的需求。

SM4 密码算法是一个分组算法，分组长度为 128 比特，密钥长度为 128 比特。加密算法与密钥扩展算法都采用 32 轮非线性迭代结构，数据解密和数据加密的算法结构相同，只是轮密钥的使用顺序相反，解密轮密钥是加密轮密钥的逆序。

本规范对轮函数 F 和算法进行了详细的介绍。轮函数 F 包括轮函数结构和合成置换 T，算法描述包括加密算法、解密算法和密钥扩展算法，并在附录中给出了运算示例。

28.1.3 SM2 椭圆曲线公钥密码算法（GM/T 0003）

GM/T 0003-2012《SM2 椭圆曲线公钥密码算法》

本规范包括 5 个部分：总则、数字签名算法、密钥交换协议、公钥加密算法、参数定义。

第 1 部分为总则，给出了 SM2 椭圆曲线公钥密码算法涉及的必要数学基础知识与相关密码技术，以帮助实现其他各部分所规定的密码机制，适用于基域为素域和二元扩域的椭圆曲线公钥密码算法。本部分主要包括域和椭圆曲线、数据类型及其转换、椭圆曲线系统

参数及其验证、密钥对的生成、公钥的验证等内容，并在附录中给出了椭圆曲线的背景知识、数论算法、曲线示例和椭圆曲线方程参数的拟随机生成及验证。

第 2 部分为数字签名算法，规定了 SM2 椭圆曲线公钥密码算法的数字签名算法，包括数字签名生成算法和验证算法，并给出了数字签名与验证示例及其相应的流程，适用于商用密码应用中的数字签名和验证，可满足多种密码应用中的身份认证和数据完整性、真实性的安全需求。本部分主要包括数字签名算法、数字签名的生成算法及流程、数字签名的验证算法及流程等内容，并在附录中给出了数字签名与验证的示例。

第 3 部分为密钥交换协议，规定了 SM2 椭圆曲线公钥密码算法的密钥交换协议，并给出了密钥交换与验证示例及其相应的流程。适用于商用密码应用中的密钥交换，可满足通信双方经过两次或三次信息传递过程，计算获取一个由双方共同决定的共享秘密密钥（会话密钥）。本部分主要包括算法参数与辅助函数、密钥交换协议及流程等内容，并在附录中给出了密钥交换及验证示例。

第 4 部分为公钥加密算法，规定了 SM2 椭圆曲线公钥密码算法的公钥加密算法，并给出了消息加解密示例和相应的流程。适用于国家商用密码应用中的消息加解密，消息发送者可以利用接收者的公钥对消息进行加密，接收者用对应的私钥进行解密，获取消息。本部分主要包括算法参数与辅助函数、加密算法及流程、解密算法及流程等内容，并在附录中给出了消息加解密示例。

第 5 部分为参数定义，规定了 SM2 椭圆曲线公钥密码算法的曲线参数，并在附录中给出了数字签名与验证、密钥交换与验证、消息加解密示例。

28.1.4 SM3 密码杂凑算法（GM/T 0004）

GM/T 0004-2012《SM3 密码杂凑算法》

该规范规定了 SM3 密码杂凑算法的计算方法和计算步骤，并给出了运算示例，适用于商用密码应用中的数字签名和验证、消息认证码的生成与验证以及随机数的生成，可满足多种密码应用的安全需求。同时，本规范还可为安全产品生产商提供产品和技术的标准定位以及标准化的参考，提高安全产品的可信性与互操作性。

SM3 算法可概述为：对长度为 n ($n < 2^{64}$) 位的消息 m ，SM3 杂凑算法经过填充和迭代压缩，生成杂凑值，杂凑值长度为 256 位。本规范从算法概述、迭代演练等方式对 SM3 算法进行描述，并通过具体示例进行剖析展示。

28.1.5 SM2 密码算法使用规范（GM/T 0009）

GM/T 0009-2012《SM2 密码算法使用规范》

本规范定义了 SM2 密码算法的使用方法，以及密钥、加密与签名等数据格式。适用于 SM2 密码算法的使用，以及支持 SM2 密码算法的设备和系统的研发和检测。

本规范介绍了 SM2 公钥和私钥的数学本质。SM2 私钥是一个大于或等于 1 且小于 $n-1$ 的整数（ n 为 SM2 算法的阶，其值参见 GM/T 0003），简记为 k ，长度为 256 位；SM2 公钥是 SM2 曲线上的一个点，由横坐标和纵坐标两个分量来表示，记为 (x, y) ，简记为 Q ，每个分量的长度为 256 位。

本规范介绍了 8 位字节串和位串之间的转换，包括位串到 8 位字节串的转换、8 位字节串到位串的转换、整数到 8 位字节串的转换、8 位字节串到整数的转换。

本规范介绍了 SM2 算法的数据格式，包括密钥数据格式、加密数据格式、签名数据格式、密钥对保护数据格式。

本规范还介绍了预处理和计算过程，包括生成密钥、加密、解密、签名、验签和密钥协商。

28.1.6 SM2 密码算法加密签名消息语法规范（GM/T 0010）

GM/T 0010-2012《SM2 密码算法加密签名消息语法规范》

本规范定义了使用 SM2 密码算法的加密签名消息语法，适用于使用 SM2 算法进行加密和签名操作时对操作结果的标准化封装。

本规范对 6 类对象 data, signedData, envelopedData, signedAndEnvelopedData, encryptedData 和 keyAgreementInfo 的标识符进行了定义。

对象标识符 OID	对象标识符定义
1.2.156.10197.6.1.4.2	SM2 密码算法加密签名消息语法规范
1.2.156.10197.6.1.4.2.1	数据类型 data
1.2.156.10197.6.1.4.2.2	签名数据类型 signedData
1.2.156.10197.6.1.4.2.3	数字信封数据类型 envelopedData
1.2.156.10197.6.1.4.2.4	签名及数字信封数据类型 signedAndEnvelopedData
1.2.156.10197.6.1.4.2.5	加密数据类型 encryptedData
1.2.156.10197.6.1.4.2.6	密钥协商类型 keyAgreementInfo

本规范对上述 6 类对象的数据类型结构进行了详细定义。

28.1.7 数字证书认证系统密码协议规范（GM/T 0014）

GM/T 0014-2012《数字证书认证系统密码协议规范》

本规范适用于电子政务/电子商务基于密码技术的数字证书认证系统的设计、建设、检测、运营及管理，规范数字证书认证系统中密码协议的标准化应用，推动数字证书认证系统密码协议的互联互通和相互认证。对于组织或机构内部使用的数字证书认证系统密码协议的建设、运营及管理，可参考使用。同时本规范还可安全产品生产商提供产品和技术的确定位和标准化的参考，提高安全产品的可信性和互操作性。

本规范涉及的相关协议指数字证书认证系统中涉及到密码技术的安全协议，特别是证书认证系统使用的有关安全协议。这些协议包括用户终端同 RA 之间的安全协议，RA 同 CA 之间的安全协议，CA 同 KM 之间的安全协议，CA 同 LDAP 服务之间的安全协议，CA 同 OCSP 服务之间的安全协议，用户同 LDAP 服务之间的安全协议，用户同 OCSP 服务之间的安全协议等。

本规范给出了与协议直接相关的格式、语法等内容，其中凡涉及 RSA 密码算法的，其密钥结构遵循 PKCS#1 规范，其封装结构遵循 PKCS#7 规范；凡涉及 SM2 算法的，其密钥结构遵照 GM/T 0009，其封装结构遵循 GM/T 0010；凡涉及对象标识符的应遵循 GM/T 0006。

另外,本规范附录 1 中给出了证书模板格式、证书作废列表格式、加密值、PKI 消息的状态码和故障信息等,附录 2 给出了 RA 与 CA 之间的相关协议说明,附录 3 给出了协议报文的实例,附录 4 给出了非实时发布证书协议流程。

28.1.8 基于 SM2 密码算法的数字证书格式规范 (GM/T 0015)

GM/T 0015-2012《基于 SM2 密码算法的数字证书格式规范》

本规范规定了基于 SM2 密码算法的数字证书和证书撤销(作废)列表的基本结构,并对数字证书和证书撤销列表中的各数据项内容进行了描述。适用于数字证书认证系统的研发、数字证书认证机构的运营以及基于数字证书的安全应用。

本规范详细介绍了数字证书格式,包括基本证书域和 TBSCertificate 的数据结构两部分内容。TBSCertificate 包含了证书结构中的前十项信息。这些信息主要有主体和颁发者的名称、主体的公钥、有效期、版本号和序列号,有些 TBSCertificate 还可以包含可选的唯一标识符项和扩展项。还介绍了证书扩展域及其数据结构,包括标准扩展和专用因特网扩展。

本规范还介绍了 CRL 格式,包括 CRL 的数据结构、TBSCertList 及其数据结构和 CRL 扩展项及其数据结构,并在附录中给出了证书的结构以及实例、证书和 CRL 内容表和数字证书编码举例。

28.1.9 通用密码服务接口规范 (GM/T 0019)

GM/T 0019-2012《通用密码服务接口规范》

本规范规定了统一的通用密码服务接口,适用于公开密钥应用技术体系下密码应用服务的开发,密码应用支撑平台的研制及检测,也可用于指导直接使用密码设备的应用系统的开发。

本规范对算法标识和数据结构、密码服务接口和密码服务接口函数进行了定义。算法标识和数据结构主要内容包括算法标识与常量定义、用户证书列表、密钥容器信息列表和证书中 DN 的结构;密码服务接口的主要内容包括环境类函数、证书类函数、密码运算类函数和消息类函数。

本规范中密码服务接口函数定义主要分为以下四类函数。

1. 环境类函数

- (1) 初始化环境: SAF_Initialize
- (2) 清除环境: SAF_Finalize
- (3) 获取接口版本信息: SAF_GetVersion
- (4) 用户登录: SAF_Login
- (5) 修改 PIN: SAF_ChangePin
- (6) 注销登录: SAF_Logout

2. 证书类函数

- (1) 添加根 CA 证书: SAF_AddPrustedRootCaCertificate
- (2) 获取根 CA 证书个数: SAF_GetRootCaCertificateCount

- (3) 获取根 CA 证书: SAF_GetRootCaCertificate
- (4) 删除根 CA 证书: SAF_RemoveRootCaCertificate
- (5) 添加 CA 证书: SAF_AddCaCertificate
- (6) 获取 CA 证书个数: SAF_GetCaCertificateCount
- (7) 获取 CA 证书: SAF_GetCaCertificate
- (8) 删除 CA 证书: SAF_RemoveCaCertificate
- (9) 添加 CRL: SAF_AddCrl
- (10) 验证用户证书: SAF_VerifyCertificate
- (11) 根据 CRL 文件获取用户证书注销状态: SAF_VerifyCertificateByCrl
- (12) 根据 OCSP 获取证书状态: SAF_GetCertificateStateByOCSP
- (13) 通过 LDAP 方式获取证书: SAF_GetCertificateFromLdap
- (14) 通过 LDAP 方式获取证书对应的 CRL: SAF_GetCrlFromLdap
- (15) 取证书信息: SAF_GetCertificateInfo
- (16) 取证书扩展信息: SAF_GetExtTypeInfo
- (17) 列举用户证书: SAF_EnumCertificate
- (18) 列举用户的密钥容器信息: SAF_EnumKeyContainerInfo
- (19) 释放列举用户证书的内存: SAF_EnumCertificateFree
- (20) 释放列举密钥容器信息的内存: SAF_EnumkeyContainerInfoFree

3. 密码运算类函数

- (1) 单块 Base64 编码: SAF_Base64_Encode
- (2) 单块 Base64 解码: SAF_Base64_Decode
- (3) 创建 Base64 对象: SAF_Base64_CreateBase64Obj
- (4) 销毁 Base64 对象: SAF_Base64_DestroyBase64Obj
- (5) 通过 Base64 对象继续编码: SAF_Base64_EncodeUpdate
- (6) 通过 Base64 对象编码结束: SAF_Base64_EncodeFinal
- (7) 通过 Base64 对象继续解码: SAF_Base64_DecodeUpdate
- (8) 通过 Base64 对象解码结束: SAF_Base64_DecodeFinal
- (9) 生成随机数: SAF_GenRandom
- (10) HASH 运算: SAF_Hash
- (11) 创建 HASH 对象: SAF_CreateHashObj
- (12) 删除 HASH 对象: SAF_DestroyHashObj
- (13) 通过对象多块 HASH 运算: SAF_HashUpdate
- (14) 结束 HASH 运算: SAF_HashFinal
- (15) 生成 RSA 密钥对: SAF_GenRsaKeyPair
- (16) 获取 RSA 公钥: SAF_GetPublicKey
- (17) RSA 签名运算: SAF_RsaSign
- (18) 对文件进行 RSA 签名运算: SAF_RsaSignFile
- (19) RSA 验证签名运算: SAF_RsaVerifySign

- (20) 对文件及其签名进行 RSA 验证: SAF_RsaVerifySignFile
- (21) 基于证书的 RSA 公钥验证: SAF_VerifySignByCert
- (22) 生成 ECC 密钥对: SAF_GenEccKeyPair
- (23) 获取 ECC 公钥: SAF_GetEccPublicKey
- (24) ECC 签名: SAF_EccSign
- (25) ECC 验证: SAF_EccVerifySign
- (26) ECC 公钥加密: SAF_EccPublicKeyEnc
- (27) 基于证书的 ECC 公钥加密: SAF_EccPublicKeyEncByCert
- (28) 基于证书的 ECC 公钥验证: SAF_EccVerifySignByCert
- (29) 创建对称算法对象: SAF_CreateSymmAlgoObj
- (30) 生成会话密钥并用外部公钥加密输出: SAF_GenerateKeyWithEPKu
- (31) 导入加密的会话密钥: SAF_ImportEncdedKey
- (32) 生成密钥协商参数并输出: SAF_GenerateAgreementDataWithECC
- (33) 计算会话密钥: SAF_GenerateKeyWithECC
- (34) 产生协商数据并计算会话密钥: SAF_GenerateAgreementDataAndKeyWithECC
- (35) 销毁对称算法对象: SAF_DestroySymmAlgoObj
- (36) 销毁会话密钥句柄: SAF_DestroyKeyHandle
- (37) 单块加密运算: SAF_SymmEncrypt
- (38) 多块加密运算: SAF_SymmEncryptUpdate
- (39) 结束加密运算: SAF_SymmEncryptFinal
- (40) 单块解密运算: SAF_SymmDecrypt
- (41) 多块解密运算: SAF_SymmDecryptUpdate
- (42) 结束解密运算: SAF_SymmDecryptFinal
- (43) 单组数据消息鉴别码运算: SAF_Mac
- (44) 多组数据消息鉴别码运算: SAF_MacUpdate
- (45) 结束消息鉴别码运算: SAF_MacFinal

4. 消息类函数

- (1) 编码 PKCS#7 格式的带签名的数字信封数据: SAF_Pkcs7_EncodeData
- (2) 解码 PKCS#7 格式的带签名的数字信封数据: SAF_Pkcs7_DecodeData
- (3) 编码 PKCS#7 格式的签名数据: SAF_Pkcs7_EncodeSignedData
- (4) 解码 PKCS#7 格式的签名数据: SAF_Pkcs7_DecodeSignedData
- (5) 编码 PKCS#7 格式的数字信封数据: SAF_Pkcs7_EncodeEnvelopedData
- (6) 解码 PKCS#7 格式的数字信封数据: SAF_Pkcs7_DecodeEnvelopedData
- (7) 编码 PKCS#7 格式的摘要数据: SAF_Pkcs7_EncodeDigestedData
- (8) 解码 PKCS#7 格式的摘要数据: SAF_Pkcs7_DecodeDigestedData
- (9) 编码基于 SM2 算法的带签名的数字信封数据: SAF_SM2_EncodeSignedAndEnvelopedData
- (10) 解码基于 SM2 算法的带签名的数字信封数据: SAF_SM2_DecodeSignedAnd

EnvelopedData

- (11) 编码基于 SM2 算法的签名数据: SAF_SM2_EncodeSignedData
- (12) 解码基于 SM2 算法的签名数据: SAF_SM2_DecodeSignedData
- (13) 编码基于 SM2 算法的数字信封: SAF_SM2_EncodeEnvelopedData
- (14) 解码基于 SM2 算法的数字信封: SAF_SM2_DecodeEnvelopedData

在附录中给出了上述接口函数返回值错误代码的定义。

28.1.10 证书应用综合服务接口规范 (GM/T 0020)

GM/T 0020-2012《证书应用综合服务接口规范》

本规范规定了面向证书应用的统一服务接口,适用于公钥密码应用技术体系下密码应用服务产品的开发,密码应用支撑平台的研制及检测,也可用于指导直接使用密码设备和密码服务应用系统的集成和开发。

证书应用综合服务接口位于应用系统和典型密码服务接口之间,向应用层直接提供证书信息解析、基于数字证书身份认证和信息的机密性、完整性、不可否认性等高级密码服务。该接口可直接供应用系统调用,将应用系统的密码服务请求转向通用密码服务接口,通过通用密码服务接口调用相应的密码设备实现具体的密码运算和密钥操作。通用密码服务接口应遵循 GM/T 0019。

本规范所定义的证书应用综合服务接口包括客户端服务接口和服务器端服务接口两类,其中服务器端服务接口采用 COM 组件形式和 Java 形式描述。

客户端服务接口采用客户端控件方式,客户端控件适用于客户端程序调用,接口的形态包括 DLL 动态库、ActiveX 控件、Applet 插件等,接口应支持 Windows XP、Windows 2000、Windows 2003、Vista、Windows 7 等终端用户使用的主流操作系统。客户端控件接口的主要函数功能应包括:配置管理、证书解析、签名与验证、加密与解密、数字信封、XML 数据的签名与验证等。在定义客户端服务接口时,本规范以 ActiveX 控件为例进行描述,其中 BSTR 代表函数返回值或参数类型为 OLECHAR 字符串类型,不同的开发语言应采取对应的类型定义,如 char*、CString、java.lang.String 等。

服务器端服务接口适用于服务器端程序调用,接口的形态包括 COM 组件、JAR 包、WebService 等形态,接口应支持 Windows、Linux、UNIX、AIX、Solaris 等服务器使用的主流操作系统。服务器端服务接口的函数功能与客户端控件接口相对应,主要包括:配置管理、数字证书解析、签名与验证、加密与解密、数据信封、XML 数据的签名与验证、时间戳等。

本规范在附录中给出了证书应用综合服务接口的错误代码返回值、典型部署模型和集成示例。

28.1.11 IPSec VPN 技术规范 (GM/T 0022)

GM/T 0022-2014 IPSec VPN 技术规范

本规范的协议部分主要依据 RFC4301、RFC4302、RFC4303、RFC4308、RFC4309 等标准制定。按照我国相关密码政策和法规,结合我国实际应用需求及产品生产厂商的实践经验,对密钥协商、密码算法及使用、某些功能项的实施方法提出了一些特定的要求。

本规范对 IPSec VPN 的技术协议、产品功能、性能和管理以及检测进行了规定,可用

于指导 IPSec VPN 产品的研制、检测、使用和管理。基于本规范研制的 IPSec VPN 产品所用的密码算法和密码部件须经国家密码管理局审批。本规范中未明确指明为可选要素的部分均为必备要素。

本规范主要由范围、规范性引用文件、术语与缩略语、密码算法和密钥种类、协议、IPSec VPN 产品要求、IPSec VPN 产品检测、合格判定等 8 章内容和附录 A（资料性附录）IPSec VPN 概述组成。其中，协议由密钥交换协议、安全报文协议等两节内容组成。IPSec VPN 产品要求由产品功能要求、产品性能要求、安全管理要求等 3 节内容组成。IPSec VPN 产品检测由产品功能检测、产品性能检测、安全管理检测等 3 节内容组成。

28.1.12 SSL VPN 技术规范（GM/T 0024）

GM/T 0024-2014 SSL VPN 技术规范

本规范的协议内容参照传输层安全协议（RFC4346 TLS1.1）。依据我国相关密码政策和法规，结合我国实际应用需求及产品生产厂商的实践经验，在 TLS1.1 的握手协议中增加了 ECC、IBC 的认证模式和密钥交换模式，取消了 DH 密钥协商方式，修改了密码套件的定义。此外，在本规范中还增加了网关-网关协议。

本规范对 SSL VPN 的密码算法、技术协议、产品的功能、性能和管理以及检测进行了规定。本规范适用于 SSL VPN 产品的研制，也可用于指导 SSL VPN 产品的检测和使用。本规范中未明确指明为可选要素的部分均为必备要素。基于本规范研制的 SSL VPN 产品所用的密码算法和密码部件须经国家密码管理局审批。

本规范主要由范围、规范性引用文件、术语和定义、符号和缩略语、密码算法和密钥种类、协议、产品要求、产品检测基本要求、合格判定等 9 章内容组成。其中，协议由概述、数据类型定义、记录层协议、握手协议族、密钥计算、网关到网关协议等 6 节内容组成。产品要求由产品功能要求、产品性能要求、安全管理要求等 3 节内容组成。产品检测基本要求由产品功能检测基本要求、产品性能检测基本指标、安全管理检测基本要求等 3 节内容组成。

28.1.13 安全认证网关产品规范（GM/T 0026）

GM/T 0026-2014《安全认证网关产品规范》

本规范规定了安全认证网关产品使用的密码算法和密钥种类，对安全认证网关产品的功能、性能、安全管理以及产品检测进行了规定，可用于指导安全认证网关产品的研制、检测、使用和管理。本规范有效解决了安全认证网关产品的安全性和规范性的统一，有利于节省产品开发商的开发成本和开发难度，有利于各厂家产品之间的互联互通。本规范有利于提升用户对产品的规范使用和管理水平，有利于相关检测机构对该类产品的规范化检测。

本规范共包括 9 章正文：

第 1 章“范围”，描述了本规范的适用范围、规范的边界等。

第 2 章“规范性引用文件”，描述了本规范应用的相关文件，包括《随机数检测规范》、《IPSec VPN 技术规范》、《SSL VPN 技术规范》、《密码设备管理规范》等规范。

第 3 章“术语和缩略语”，介绍了规范中使用的术语和缩略语。

第4章“符号和缩略语”，介绍了规范中使用的符号和缩略语。

第5章“安全认证网关概述”，给出了安全认证网关产品的概述。

第6章“密码算法和密钥种类”，列出了规范中使用的密码算法和密钥种类。

第7章“安全认证网关产品要求”，从产品功能、产品性能、安全性要求和管理要求等方面对安全认证网关产品提出了具体要求。

第8章“安全认证网关产品检测”，规定了安全认证网关产品须完成的检测。第7章和第8章为本规范的关键章节。

第9章“合格判定”，说明了产品的合格判定标准。

28.1.14 签名验签服务器技术规范（GM/T 0029）

GM/T 0029-2014《签名验签服务器技术规范》

本标准的主要目的是规定签名验签服务器的相关技术规范，包括签名验签服务器的功能要求、安全要求、接口要求、检测要求和消息协议语法规则等有关内容，为签名验签服务器的研制和开发提供指导和依据。

本规范共包括8章正文和2个附录：第1章“范围”；第2章“规范性引用文件”；第3章“术语和定义”；第4章“符号和缩略语”；第5章“签名验签服务器的功能要求”；第6章“签名验签服务器的安全要求”；第7章“签名验签服务器的检测要求”；第8章“合格判定”；附录A“消息协议语法规则”；附录B“响应码定义和说明”。

其中，第5章描述了签名验签服务器的主要功能，包括初始化功能要求、与CA基础设施的连接功能要求、应用管理功能要求、证书管理和验证功能要求、数字签名功能要求、访问控制功能要求、设备管理功能要求、日志管理功能要求、设备自检功能要求。

第6章定义了签名验签服务器在密码设备、系统要求、使用要求、管理要求、物理安全、网络部署、API接口、环境适应性、可靠性等方面的要求。

第7章定义了签名验签服务器的检测要求，包括外观和结构、提交文档、功能检测、性能检测、环境适应性检测等检测内容。

附录A中签名验签服务的消息协议接口采用请求响应模式，协议模型由请求者、响应者和它们之间的交互协议组成。通过本协议，请求者将数字签名、验证数字签名等请求发送给响应者，由响应者完成签名验签服务并返回结果。本规范中的接口消息协议包括导出证书、解析证书、验证证书有效性、数字签名、验证数字签名、消息签名、验证消息签名等服务功能。

附录B定义了消息协议的响应码。

28.1.15 安全电子签章密码技术规范（GM/T 0031）

GM/T 0031-2014《安全电子签章密码技术规范》

本规范的制定可以有效提升电子签章产品使用密码技术的安全性和规范性，保障行业的健康发展。同时，有利于节省产品开发商的开发成本和开发难度，有利于各厂家产品之间的互联互通，降低用户选用产品的技术门槛，提升用户对产品的规范使用和管理水平，有利于相关检测机构对该类产品的规范化检测。

本规范共包括6章正文：第1章“范围”；第2章“规范性引用文件”；第3章“术语和定义”；第4章“符号和缩略语”；第5章“电子签章的密码应用安全机制”；第6章“电

子签章密码应用协议”。

其中，第1章至第4章为总述性内容，介绍规范的范围、规范引用、关键术语等。

第5章和第6章为电子签章密码应用的关键章节，详细规定了数据格式、密码处理流程等。

第5章介绍了电子签章密码应用的基本安全机制。

第6章分别对电子印章和电子签章的数据格式、处理流程等，从密码安全应用的角度进行了规范。

28.1.16 时间戳接口规范（GM/T 0033）

GM/T 0033-2014《时间戳接口规范》

本标准的目标是在上层应用与时间戳系统之间制定一套统一的时间戳服务接口，为应用系统提供精确可信的时间认证服务，确保时间戳服务在信息系统的应用中能够通用，实现信息系统互联互通，为应用实现业务处理的抗抵赖性提供基础，同时有利于相关检测机构对该类产品的规范化检测。遵循本规范的时间戳服务产品可以实现和时间戳系统无关以及和证书认证系统无关的时间认证服务，保证时间戳服务对用户、对应用的透明性和无关性。

本规范共包括9章正文：第1章“范围”；第2章“规范性引用文件”；第3章“术语和定义”；第4章“符号和缩略语”；第5章“标识和数据结构”；第6章“时间戳服务描述”；第7章“时间戳的请求和响应格式”；第8章“时间戳服务与时间戳服务机构的通讯方式”；第9章“时间戳服务接口组成和功能说明”。

其中，第1章至第4章为总述性内容，介绍规范的整体情况、关键术语、符号和缩略语等。

第5章介绍了规范中定义的算法标识和数据结构，包括标识定义、密码服务接口以及时间戳服务接口常量定义。

第6章描述了时间戳服务接口的逻辑结构。

第7章定义了时间戳服务请求格式、响应格式的ASN.1编码方式。

第8章定义了5种通讯方式：电子邮件方式、文件方式、Socket方式、HTTP方式、SOAP方式，并规定了消息传输格式。

第9章描述了7个与时间戳服务有关的函数，涵盖了获取时间戳服务的全部功能，包括环境函数和时间戳服务函数两大类。环境函数类中包含初始化环境和清除环境函数，时间戳服务函数类中包含生成时间戳请求、生成时间戳应答、验证时间戳有效性、获取时间戳主要信息、解析时间戳详细信息。在上述部分中详细描述了各个接口函数，包括各函数的原型、描述、参数说明、返回值说明及备注等详细信息。

28.1.17 基于 SM2 密码算法的证书认证系统密码及其相关安全技术规范（GM/T 0034）

GM/T 0034-2014《基于 SM2 密码算法的证书认证系统密码及其相关安全技术规范》

本标准主要是规定了为公众服务的数字证书认证系统的设计、建设、检测、运行及管理规范。本规范的目标是为实现数字证书认证系统的互联互通和交叉认证提供统一的依据，

指导第三方认证机构的数字证书认证系统的建设和检测评估，规范数字证书认证系统中密码及相关安全技术的应用，并有利于相关检测机构对该类产品的规范化检测。同时非第三方认证机构的数字证书认证系统的建设、运行及管理也可参照本标准执行。

本规范共包括 11 章正文：第 1 章“范围”；第 2 章“规范性引用文件”；第 3 章“术语”；第 4 章“缩略语”；第 5 章“证书认证系统”；第 6 章“密钥管理中心”；第 7 章“密码算法、密码设备及接口”；第 8 章“证书认证中心建设”；第 9 章“密钥管理中心建设”；第 10 章“证书认证中心运行管理要求”；第 11 章“密钥管理中心运行管理要求”。

其中，第 1 章至第 4 章为总述性内容，介绍规范的整体情况、关键术语、符号和缩略语等。

第 5 章介绍了证书认证系统的设计细节，包括系统的总体设计和各子系统设计，并提供了设计原则以及各个子系统的实现方式。

第 6 章描述了密钥管理中心的组成模块，包括密钥生成、密钥管理、密钥库管理、认证管理、安全审计、密钥恢复和密码服务等模块。

第 7 章定义了证书认证系统的密码算法、密码设备和接口。

第 8 章从系统、安全、数据备份、可靠性、物理安全、人事管理制度等方面规范了证书认证中心的建设。

第 9 章从系统、安全、数据备份、可靠性、物理安全、人事管理制度等方面规范了密钥管理中心的建设。

第 10 章从人员管理、业务运行管理、密钥分管、安全管理、安全审计、文档配备等方面对证书认证中心的运行管理要求进行了规范。

第 11 章从人员管理、业务运行管理、密钥分管、安全管理、安全审计、文档配备等方面对密钥管理中心的运行管理要求进行了规范。

28.1.18 证书认证系统检测规范（GM/T 0037）

GM/T 0037-2014《证书认证系统检测规范》

本规范的编制主要用于检测按照《证书认证系统密码及其相关安全技术规范》研制或建设的数字证书认证系统，也可以为其他数字证书认证系统的研制、建设提供参考作用。其主要作用是作为国内第三方认证服务系统提供一个标准的、规范的测试范围、测试内容、测试方法以及判定策略。

本规范主要由以下几部分对数字证书认证系统的检测进行规范。

1. 检测对象

《规范》确定适用的检测对象为证书认证系统产品及按《证书认证系统密码及其相关安全技术规范》要求建设的证书认证服务运营系统项目。

2. 测试大纲

规定了测试大纲编制的原则，并在附录中提供了可供参考的测试大纲的示例。

3. 检测环境

确定了产品和项目的检测环境。

4. 检测内容

对场地、网络、岗位及权限管理、安全管理、系统初始化、系统功能、性能、数据备份和恢复、第三方安全产品、入根、证书格式、证书链、算法等 13 个方面详细规定了检测内容。

5. 检测方法

规定了 13 项检测内容的具体检测方法。

6. 合格判定

规定了判定产品和项目合格的最低条件。

通过检测内容和检测方法可以很清楚地了解到研制、建设一个第三方认证服务系统时应具备的系统功能和检测内容。规范最后还附有检测大纲、认证系统的网络结构、认证系统机房布局和物理连线等资料性附录，供国内 PKI 厂商和运营商参考。

28.1.19 证书认证密钥管理系统检测规范（GM/T 0038）

GM/T 0038-2014《证书认证密钥管理系统检测规范》

《规范》的编制主要用于检测按照《证书认证系统密码及其相关安全技术规范》研制或建设的密钥管理系统，也可以为其他密钥管理系统的研制、建设提供参考。其主要作用是作为国内第三方认证服务系统的密钥管理系统提供一个标准的、规范的测试范围、测试内容、测试方法以及判定策略。

本规范主要由以下几部分对密钥管理系统的检测进行了规范。

1. 检测对象

《规范》确定适用的检测对象为密钥管理系统产品及按《证书认证系统密码及其相关安全技术规范》要求建设的密钥管理系统项目。

2. 测试大纲

规定了测试大纲编制的原则，并在附录中提供了可供参考的测试大纲的示例。

3. 检测环境

确定了产品和项目的检测环境。

4. 检测内容

对场地、网络、岗位及权限管理、安全管理、系统初始化、系统功能、性能、数据备份和恢复、第三方安全产品等 9 个方面详细规定了检测内容。

5. 检测方法

规定了 9 项检测内容的具体检测方法。

6. 合格判定

规定了判定产品和项目合格的最低条件。

通过检测内容和检测方法可以很清楚地了解到研制、建设一个密钥管理系统时应具备的系统功能和检测内容。规范最后还附有检测大纲、密钥管理系统的网络结构、密钥管理

系统机房布局和物理连线等资料性附录，供国内 PKI 厂商和运营商参考。

28.1.20 证书认证系统密码及其相关安全技术规范（GB/T 25056）

GB/T 25056-2010《信息安全技术 证书认证系统密码及其相关安全技术规范》

本标准适用于在中华人民共和国境内建设并正式运行的、为公众服务的数字证书认证系统的设计、建设、检测、运行及管理，为实现数字证书认证系统的互联互通和交叉认证提供统一的依据，指导数字证书认证系统的建设和检测评估，规范数字证书认证系统中密码及相关安全技术的应用。其他数字证书认证系统的建设、运行及管理，可参照本标准。

本标准主要由范围、规范性引用文件、术语和缩略语、证书认证系统、密钥管理系统、密钥算法/密码设备及接口、协议、证书认证中心建设、密钥管理中心建设、证书认证中心运行管理要求、密钥管理中心运行管理要求、检测等 12 章内容和附录 A（资料性附录）KMC 与 CA 之间的消息格式、附录 B（资料性附录）安全通信协议、附录 C（资料性附录）密码设备接口函数定义及说明、附录 D（资料性附录）证书认证系统网络结构图、附录 E（资料性附录）证书申请和下载格式等组成。

其中，证书认证系统由功能描述、系统设计、数字证书、证书注销列表等 4 节内容组成。密钥管理系统由功能描述、系统设计、KMC 与 CA 的安全通信协议等 3 节内容组成。密码算法/密码设备及接口由密码算法、密码设备、密码服务接口等 3 节内容组成。协议由证书管理协议、证书验证协议、安全通信协议等 3 节内容组成。证书认证中心建设由系统、安全、数据备份、可靠性、物理安全、人事管理制度等 6 节内容组成。密钥管理中心建设由系统、安全、数据备份、可靠性、物理安全、人事管理制度等 6 节内容组成。证书认证中心运行管理要求由人员管理要求、CA 业务运行管理要求、密钥分管要求、安全管理要求、安全审计要求、文档的配备等 6 节内容组成。密钥管理中心运行管理要求由人员管理要求、运行管理要求、密钥分管、安全管理、安全审计、文档配备等 6 节内容组成。检测由系统初始化、用户注册管理系统、证书/证书注销列表生成与签发系统、证书/证书注销列表存储与发布系统、证书状态查询系统、安全审计系统、密钥管理系统检测、系统安全性检测、其他安全产品和系统等 9 节内容组成。附录 A 由概述、协议等 2 节内容组成。附录 B 由符号说明、身份认证、密钥交换、安全通信协议等 4 节内容组成。附录 C 由应用类密码设备接口函数、证书载体接口函数等 2 节内容组成。附录 D 由当 RA 采用 C/S 模式时 CA 的网络接口、当 RA 采用 B/S 模式时 CA 的网络接口、CA 与远程 RA 的连接、KMC 与多个 CA 的网络连接等 4 节内容组成。附录 E 由证书申请格式、证书下载格式等 2 节内容组成。

28.1.21 电子认证服务机构运营管理规范（GB/T 28447）

GB/T 28447-2012《信息安全技术 电子认证服务机构运营管理规范》

本标准规定了电子认证服务机构在业务运营、认证系统运行、物理环境与设施安全、组织与人员管理、文档、记录、介质管理、业务连续性、审计与改进等多方面应遵循的要求。本标准适用于在开放互联环境中提供数字证书服务的电子认证服务机构的建设、管理及评估。对于在封闭环境中（如在特定团体或某个行业内）运行的电子认证服务机构可根据自身安全风险评估以及国家有关的法律法规有选择性地参考本标准。国家有关的测评机

构、监管部门也可以将本标准作为测评和监管的依据。

本标准主要由范围、规范性引用文件、术语和定义、缩略语、电子认证服务机构运营的业务、业务运营中的风险、认证系统运行要求、物理环境与设施、组织与人员管理、文档/记录与介质管理、业务连续性要求、审计与改进等 12 章内容和附录 A（资料性附录）业务运营风险举例等组成。

其中，电子认证服务机构运营的业务由用户证书服务、用户证书密钥服务、认证系统功能要求、认证业务流程要求等 4 节内容组成。认证系统运行要求由网络安全、主机系统安全、系统冗余与备份、系统运营维护安全管理、密码设备安全管理、CA 密钥和证书管理等 6 节内容组成。物理环境与设施由运营场地、运营区域划分及要求、安全监控系统、环境保护与控制设施、支撑设施、场地访问安全管理、场地监控安全管理、注册机构场地安全等 8 节内容组成。组织与人员管理由职能与角色设置、安全组织、人员安全管理等 3 节内容组成。业务连续性要求由业务连续性计划、应急处理预案、灾难恢复计划、灾备中心等 4 节内容组成。

28.2 行业性标准

28.2.1 卫生系统电子认证服务规范

《卫生系统电子认证服务规范》（试行），2010 年 4 月

本规范依据《卫生系统电子认证服务管理办法（试行）》，定义了参与卫生系统电子认证服务体系建设的管理方、使用方和提供方开展电子认证服务的工作机制，描述了卫生系统电子认证服务的总体要求，规范了电子认证服务机构需要遵循的证书业务服务和证书支持服务的要求，提出了服务的保障要求。本规范适用于卫生系统电子认证服务体系建设的管理方、使用方和提供方。

本规范共包括 5 章正文和 1 个附录：第 1 章“范围”；第 2 章“服务总体要求”；第 3 章“证书业务服务”；第 4 章“技术支持服务”；第 5 章“服务的保障”；附录（资料性）“名词解释”。

其中，第 2 章包括证书分类、证书产品、服务模式、服务内容、平台接入等要求。卫生系统数字证书共分 6 类：内部机构证书、内部工作人员证书、内部设备证书、外部机构证书、外部个人证书、外部设备证书。电子认证服务机构交付给卫生系统用户的证书产品应包括：证书介质、证书初始保护口令、证书使用说明书以及证书管理工具等。证书服务模式包括 3 类：集中服务模式、多级服务模式、认证机构直接服务模式。电子认证服务机构提供的服务包括证书业务服务和技术支持服务两部分内容。其中，证书业务服务是指电子认证服务机构为卫生系统用户提供的证书申请、发放、更新、吊销、解锁、密钥恢复、证书查询等证书业务办理服务；技术支持服务是指电子认证服务机构在用户使用证书过程中提供的各种方式的咨询、培训、应急等服务内容。电子认证服务机构在卫生系统开展服务前，须接入卫生部数字证书服务管理平台，遵循卫生系统数字证书服务管理平台的数据同步接口要求，实现数字证书和黑名单的同步。

第 3 章规定了电子认证服务机构应为卫生系统用户提供的多种证书业务服务，包括证

书申请、证书发放、证书更新、证书吊销、证书解锁、密钥恢复和证书查询等。

第4章规定了电子认证服务机构应向卫生系统用户通过热线电话、网站、现场等服务方式,提供使用帮助、应用咨询培训、应急保障和应用集成支持等一系列技术支持服务。

第5章规定了电子认证服务机构的组织保障、制度保障、安全保障等要求。

28.2.2 卫生系统数字证书应用集成规范

《卫生系统数字证书应用集成规范》(试行), 2010年4月

本规范依据《卫生系统电子认证服务管理办法(试行)》,参照国家密码管理局“公钥密码基础设施应用技术体系”系列技术规范,结合卫生系统业务特点,提出卫生系统数字证书应用集成目标、集成要求、集成内容,定义统一的证书应用接口,并提供证书应用接口的典型部署示例、登录认证流程示例和签名验证流程示例。

本规范用于指导并规范卫生信息系统证书应用集成实施工作,指导电子认证服务机构开发标准统一的证书应用接口,规范卫生信息系统实现基于数字证书的安全登录、数字签名和加密解密等安全功能。

本规范共包括5章正文和2个附录:第1章“范围”;第2章“应用集成目标”;第3章“应用集成要求”;第4章“应用集成内容”;第5章“统一证书应用接口规范”;附录A(资料性)“证书应用接口实施示例”;附录B(资料性)“名词解释”。

其中,第4章,规定了卫生信息系统证书应用集成的5个方面的内容,包括:基于数字证书的身份认证、数字签名和验证、数据加密和解密、时间戳应用、密码设备应用等。

第5章介绍了统一证书应用接口和证书应用综合服务接口,并给出了证书应用综合服务接口的客户端接口函数定义、服务端COM组件函数定义、服务端Java函数定义等。

统一证书应用接口位于应用系统和密码设备之间,包括:证书介质应用接口、密码设备应用接口、通用密码服务接口和证书应用综合服务接口。证书应用综合服务接口是供应用系统直接调用的高级证书应用接口,一般情况下分成客户端接口和服务器端接口两个模块。

客户端接口是供应用系统客户端程序直接调用的高级接口,应支持所有主流操作系统。客户端接口应支持符合《卫生系统数字证书格式规范》的数字证书,支持使用符合《卫生系统数字证书介质技术规范》的证书介质。客户端接口可包括DLL动态库、ActiveX控件、Applet插件等多种产品形态,支持B/S和C/S等架构的应用系统,客户端接口还包括32个主要函数。

服务器端接口是供应用系统服务器端程序直接调用的高级接口,应支持所有主流操作系统,支持B/S和C/S等系统架构,支持符合《卫生系统数字证书格式规范》的数字证书,可通过添加证书信任列表的方式实现不同电子认证服务机构证书之间的交叉认证和互信互认。服务器端接口可包括COM组件、Java组件等多种形态。服务器端接口还包括36个主要函数。

28.2.3 卫生系统数字证书格式规范

《卫生系统数字证书格式规范》(试行), 2010年4月

本规范描述了卫生系统电子认证服务体系中使用的数字证书的类型、数字证书和证书撤销列表的格式,制定了各类数字证书及证书撤销列表格式模板,用于指导电子认证服务

机构签发统一格式的数字证书和证书撤销列表，以保障数字证书在卫生系统内各信息系统之间的互信互认。

本规范在 GB/T 20518-2006《信息安全技术 公钥基础设施 数字证书格式》基础上，针对卫生系统的电子认证需求，在证书类型、证书格式模板、实体唯一标识扩展项等方面进行了扩充，以适应卫生系统的业务特点和应用需求。

本规范共包括 5 章正文和 3 个附录：第 1 章“范围”；第 2 章“数字证书类别”；第 3 章“数字证书基本格式”；第 4 章“数字证书模板”；第 5 章“CRL 格式”；附录 A（资料性）“证书主题 DN 命名示例”；附录 B（资料性）“证书格式编码示例”；附录 C（资料性）“名词解释”。

其中，第 2 章根据卫生系统用户特点及应用需求，数字证书按照内部用户和外部用户分成如下 6 类：内部机构证书、内部工作人员证书、内部设备证书、外部机构证书、外部个人证书、外部设备证书。

第 3 章规定了证书格式的基本结构、基本证书域、签名算法域、签名值域、命名规范。

第 4 章规定了 3 种证书模板：个人证书模板、机构证书模板、设备证书模板。

第 5 章规定了 CRL 的基本结构和模板。

28.2.4 卫生系统数字证书介质技术规范

《卫生系统数字证书介质技术规范》（试行），2010 年 4 月

本规范描述了在卫生系统中使用的数字证书介质的各项要求，包括对安全机制、软件要求、硬件技术指标要求等内容，用于指导电子认证服务机构在卫生系统内进行数字证书介质的定制和选型。

本规范共包括 4 章正文和 1 个附录：第 1 章“范围”；第 2 章“安全机制”；第 3 章“软件要求”；第 4 章“硬件技术指标”；附录（资料性）“名词解释”。

其中，第 2 章规定了证书介质初始化、证书介质的安装注册、介质口令管理、扩展区要求、介质类型要求等过程中具体的安全机制要求。

第 3 章规定了证书介质接口、安装程序、卸载程序过程中具体的软件要求。证书介质接口函数应遵守国家密码管理局《智能 IC 卡及智能密码钥匙密码应用接口规范》的函数定义和数据结构定义，提供设备管理、访问控制、文件管理和密码服务等相关密码服务接口。

设备管理主要完成设备的等待、设备插拔、枚举、连接、断开、设置设备标签、获取设备信息、锁定设备、解锁设备操作，并定义了 9 个设备管理类接口函数。

访问控制主要完成设备认证，修改设备认证密钥，校验 PIN，修改 PIN，解锁 PIN 和清除安全状态操作，并定义了 7 个访问控制接口函数。

文件管理函数用以满足用户扩展开发的需要，包括创建文件，删除文件，枚举文件，获取文件信息，文件读写操作，并定义了 6 个文件管理接口函数。

密码服务函数提供对称算法运算、非对称算法运算、杂凑运算、消息鉴别码计算等功能，并定义了 42 个密码服务接口函数。

28.2.5 卫生系统数字证书服务管理平台接入规范

《卫生系统数字证书服务管理平台接入规范》(试行), 2010年4月

根据《卫生系统电子认证服务管理办法(试行)》相关要求,电子认证服务机构在开展服务前须接入卫生部数字证书服务管理系统。本规范描述了电子认证服务机构的CA系统(简称CA系统)接入卫生部数字证书服务管理系统的总体要求、CA系统功能要求以及CA系统接入接口要求等。

本规范用于指导相关电子认证服务机构将CA系统接入卫生部数字证书服务管理系统,实现系统接入过程标准化及安全控制,实现数字证书服务的统一管理。

本规范共包括5章正文和1个附录:第1章“范围”;第2章“系统接入总体要求”;第3章“CA系统功能要求”;第4章“CA系统接入接口要求”;第5章“WSDL文件”;附录(资料性)“名词解释”。

其中,第2章规定,根据《卫生系统电子认证服务管理办法(试行)》的规定,卫生部将建设集中的数字证书服务管理系统,用于卫生系统内所有证书用户信息的收集、查询、统计和分析,以及进行用户意见收集、服务质量监督等管理工作。卫生部通过数字证书服务管理系统对在卫生系统领域开展电子认证服务的CA机构实行接入控制及服务管理。拟为卫生系统领域提供服务的电子认证服务机构,须符合《卫生系统电子认证服务管理办法(试行)》的相关要求,将CA系统接入到卫生部数字证书服务管理系统。

第3章规定,电子认证服务机构的CA系统须具备如下基本功能:证书申请、证书更新、证书解锁、证书吊销、密钥恢复、证书信息发布。

第4章规定,电子认证服务机构的CA系统调用数字证书服务管理系统提供的数据同步接口,实现与卫生部数字证书服务管理系统之间数字证书和黑名单的信息同步等,并定义了3个数据同步接口函数原型:证书信息同步接口函数、黑名单信息同步接口函数、查询证书信息接口函数。

28.2.6 网上银行系统信息安全通用规范(JR/T 0068)

JR/T 0068-2012《网上银行系统信息安全通用规范》

本规范包含网上银行系统的描述、安全技术规范、安全管理规范、业务运作安全规范,本规范适用于网上银行系统建设、运营及测评。

本规范共包括引言、6章正文和3个附录:第1章“范围”;第2章“规范性引用文件”;第3章“术语和定义”;第4章“符号和缩略语”;第5章“网上银行系统概述”;第6章“安全规范”;附录A(资料性)“基本的网络防护架构参考图”;附录B(资料性)“增强的网络防护架构参考图”;附录C(规范性)“物理安全”。

其“引言”中指出,本规范是在收集、分析评估检查发现的网上银行信息安全问题和已发生过的网上银行案件的基础上,有针对性地提出的安全要求,内容涉及网上银行系统的技术、管理和业务运作三个方面。本规范分为基本要求和增强要求两个层次。基本要求为最低安全要求,增强要求为本规范下发之日起的三年内应达到的安全要求,各单位应在遵照执行基本要求的同时,按照增强要求,积极采取改进措施,在规定期限内达标。本规范旨在有效增强现有网上银行系统的安全防范能力,促进网上银行规范、健康发展。本规范

范既可作为网上银行系统建设和改造升级的安全性依据，也可作为各单位开展安全检查和内部审计的依据。

第 5 章包括 4 节内容：系统标识、系统定义、系统描述、安全性描述。系统描述又分为 3 小节：客户端（含专用安全设备）、通信网络、服务器端。

第 6 章包括 3 节内容：安全技术规范、安全管理规范、业务运作安全管理。安全技术规范又分为 4 小节：客户端安全（客户端程序、客户端环境安全），专用安全设备安全（USB Key、文件证书、OTP 令牌、动态密码卡、其他专用安全设备），网络通信安全（通讯协议、安全认证），服务器端安全（物理安全、网络安全、主机安全、应用安全、数据安全及备份恢复）。安全管理规范又分为 6 小节：安全管理机构、安全策略、管理制度、人员安全管理、系统建设管理、系统运维管理。业务运作安全规范又分为 3 小节：业务申请及开通、业务安全交易机制（身份认证、交易流程、交易监控）、客户教育及权益保护。

附录 A 将网上银行分为 7 个安全区域：外联区、DMZ 区、生产应用区、生产数据区、管理区域、测试区域、系统互联区域等。

附录 C 从以下 10 个方面规范了物理安全的基本要求：物理位置的选择、物理访问控制、防盗窃和防破坏、防雷击、防火、防水和防潮、防静电、温湿度控制、电力供应、电磁保护。

第 29 章 国际 标准

29.1 PKCS 系列

PKCS 是公钥密码标准 (Public Key Cryptography Standards) 的缩写, 它是由美国 RSA 实验室与遍布全球的安全系统开发者一起合作制定的一组规范, 以推动公钥密码的发展。最早发布的 PKCS 文档是早期一群公钥技术使用者在 1991 年召开的一次会议的成果; 目前 PKCS 规范已被广泛引用和实施, 部分 PKCS 规范已经成为多个国际组织正式或事实上的标准, 如 ANSI X9 文档系列、PKIX、SET、S/MIME、SSL 等。PKCS 系列主要包括以下标准。

- PKCS #1: RSA Cryptography Standard (RSA 密码标准);
- PKCS #2: 已撤销, 用以规范 RSA 加密摘要的转换方式, 已并入 PKCS#1;
- PKCS #3: Diffie-Hellman Key Agreement Standard (DH 密钥协商标准);
- PKCS #4: 已撤销, 用以定义 RSA 密钥的格式, 已并入 PKCS #1;
- PKCS #5: Password-Based Cryptography Standard (基于口令的密码标准);
- PKCS #6: Extended-Certificate Syntax Standard (扩展的证书语法标准);
- PKCS #7: Cryptographic Message Syntax Standard (密码消息语法标准);
- PKCS #8: Private-Key Information Syntax Standard (私钥信息语法标准);
- PKCS #9: Selected Attribute Types (可供选择的属性类型);
- PKCS #10: Certification Request Syntax Standard (证书请求语法标准);
- PKCS #11: Cryptographic Token Interface Standard (密码 Token 接口标准);
- PKCS #12: Personal Information Exchange Syntax Standard (个人信息交换语法标准);
- PKCS #13: Elliptic Curve Cryptography Standard (椭圆曲线密码标准), 正在制定中;
- PKCS #14: Pseudo-random Number Generation (伪随机数生成算法 PRNG), 正在制定中;
- PKCS #15: Cryptographic Token Information Format Standard (密码 Token 信息格式标准)。

1. PKCS #1: RSA Cryptography Standard (RSA 密码标准)

PKCS #1 v2.1 定义了基于 RSA 公钥算法的加密解密和签名验签机制, 其最新跟踪标准为 RFC 3447, 主要包括以下内容。

① 密钥类型: 包括公钥和私钥。

RSA 公钥格式用 ASN.1 描述如下:

```
RSAPublicKey ::= SEQUENCE {  
    modulus          INTEGER, -- n  
    publicExponent   INTEGER  -- e  
}
```

RSA 私钥格式用 ASA.1 描述如下:

```

RSAPrivateKey ::= SEQUENCE {
    version          Version,
    modulus           INTEGER,  -- n
    publicExponent    INTEGER,  -- e
    privateExponent   INTEGER,  -- d
    prime1            INTEGER,  -- p
    prime2            INTEGER,  -- q
    exponent1         INTEGER,  -- d mod (p-1)
    exponent2         INTEGER,  -- d mod (q-1)
    coefficient        INTEGER,  -- (inverse of q) mod p
    otherPrimeInfos    OtherPrimeInfos OPTIONAL
}

```

② 数据转换原子操作：包括 I2OSP（Integer-to-Octet-String primitive）和 OS2IP（Octet-String-to-Integer primitive）两种。

③ 密码原子操作：包括加密 RSAEP、解密 RSADP、签名 RSASPI 和验签 RSAVPI 等 4 种。

④ 加密解密方案：包括 RSAES-OAEP 和 RSAES-PKCS-v1_5 等两种。

⑤ 签名验签方案：包括 RSASSA-PSS 和 RSASSA-PKCS-v1_5 等两种。

⑥ 签名编码方法：包括 EMSA-PSS 和 EMSA-PKCS-v1_5 等两种。

2. PKCS #3: Diffie-Hellman Key Agreement Standard（DH 密钥协商标准）

PKCS #3 v1.4 描述了一种基于 DH 算法进行密钥协商的方法。无需预先沟通，交易双方就可以协商出一个只有双方知道的秘密密钥，该密钥可以对后续双方的数据通信进行加密保护。

3. PKCS #5: Password-Based Cryptography Standard（基于口令的密码标准）

PKCS #5 v2.0 描述了一种基于口令产生对称密钥的方法。使用 MD2 或 MD5 从口令中派生密钥，并采用 DES 的 CBC 模式加密。这个功能主要用于加密从一个计算机传送到另一个计算机的私人密钥，而不是用于加密消息。该方法在 RFC 2898 中重新发布，主要包括以下内容：

① 密钥获取函数：包括 PBKDF1 和 PBKDF2。

② 加密解密方案：包括 PBES1 和 PBES2。

③ 消息认证方案：包括 MAC 产生和 MAC 验证等。

4. PKCS #6: Extended-Certificate Syntax Standard（扩展的证书语法标准）

PKCS #6 v1.5 描述扩展证书的语法格式。该扩展证书只是对 X.509 证书格式进行了扩展，并兼容 X.509 证书格式。扩展证书格式用 ASN.1 描述如下：

```

ExtendedCertificateOrCertificate ::= CHOICE {
    certificate Certificate, -- X.509
    extendedCertificate [0] IMPLICIT ExtendedCertificate
}
ExtendedCertificate ::= SEQUENCE {

```

```

extendedCertificateInfo ExtendedCertificateInfo,
signatureAlgorithm SignatureAlgorithmIdentifier,
signature Signature }
SignatureAlgorithmIdentifier ::= AlgorithmIdentifier
Signature ::= BIT STRING
ExtendedCertificateInfo ::= SEQUENCE {
    version Version,
    certificate Certificate,
    attributes Attributes }
Version ::= INTEGER
Attributes ::= SET OF Attribute

```

5. PKCS #7: Cryptographic Message Syntax Standard (密码消息语法标准)

PKCS #7 v1.5 描述了密码消息的通用语法。该语法允许嵌套，如一个数字信封可以包含另一个数字信封，或可以对已做数字信封的数据进行签名；该语法也允许扩展各种属性，还可以用于分发证书和 CRL。PKCS #7 与 PEM 兼容，可以直接将加密的消息转换成 PEM 消息，反之亦然。PKCS #7 支持多种基于证书的管理系统，PEM 就是其中之一。在 RFC 5652 中有增强定义。该标准主要包括消息通用语法和 6 种内容类型（明文、签名、信封、签名信封、摘要、密文）。

① 消息通用语法用 ASN.1 描述如下：

```

ContentInfo ::= SEQUENCE {
    contentType ContentType,
    content
[0] EXPLICIT ANY DEFINED BY contentType OPTIONAL }
ContentType ::= OBJECT IDENTIFIER

```

② 明文消息内容用 ASN.1 描述如下：

```
Data ::= OCTET STRING
```

③ 签名消息内容用 ASN.1 描述如下：

```

SignedData ::= SEQUENCE {
    version Version,
    digestAlgorithms DigestAlgorithmIdentifiers,
    contentInfo ContentInfo,
    certificates [0] IMPLICIT ExtendedCertificatesAndCertificates
OPTIONAL,
    Crls [1] IMPLICIT CertificateRevocationLists OPTIONAL,
    signerInfos SignerInfos }
DigestAlgorithmIdentifiers ::= SET OF DigestAlgorithmIdentifier
SignerInfos ::= SET OF SignerInfo
SignerInfo ::= SEQUENCE {
    version Version,
    issuerAndSerialNumber IssuerAndSerialNumber,

```

```

digestAlgorithm DigestAlgorithmIdentifier,
authenticatedAttributes [0] IMPLICIT Attributes OPTIONAL,
digestEncryptionAlgorithm DigestEncryptionAlgorithmIdentifier,
encryptedDigest EncryptedDigest,
unauthenticatedAttributes [1] IMPLICIT Attributes OPTIONAL }
EncryptedDigest ::= OCTET STRING
DigestInfo ::= SEQUENCE {
    digestAlgorithm DigestAlgorithmIdentifier,
    digest Digest }
Digest ::= OCTET STRING

```

④ 信封消息内容用 ASN.1 描述如下：

```

EnvelopedData ::= SEQUENCE {
    version Version,
    recipientInfos RecipientInfos,
    encryptedContentInfo EncryptedContentInfo }
RecipientInfos ::= SET OF RecipientInfo
EncryptedContentInfo ::= SEQUENCE {
    contentType ContentType,
    contentEncryptionAlgorithm
    ContentEncryptionAlgorithmIdentifier,
    encryptedContent [0] IMPLICIT EncryptedContent OPTIONAL }
EncryptedContent ::= OCTET STRING
RecipientInfo ::= SEQUENCE {
    version Version,
    issuerAndSerialNumber IssuerAndSerialNumber,
    keyEncryptionAlgorithm
    KeyEncryptionAlgorithmIdentifier,
    encryptedKey EncryptedKey }
EncryptedKey ::= OCTET STRING

```

⑤ 签名信封消息内容用 ASN.1 描述如下：

```

SignedAndEnvelopedData ::= SEQUENCE {
    version Version,
    recipientInfos RecipientInfos,
    digestAlgorithms DigestAlgorithmIdentifiers,
    encryptedContentInfo EncryptedContentInfo,
    certificates [0] IMPLICIT ExtendedCertificatesAndCertificates
OPTIONAL,
    Crls [1] IMPLICIT CertificateRevocationLists OPTIONAL,
    signerInfos SignerInfos }

```

⑥ 摘要消息内容用 ASN.1 描述如下：

```

DigestedData ::= SEQUENCE {

```



```

    version Version,
    digestAlgorithm DigestAlgorithmIdentifier,
    contentInfo ContentInfo,
    digest Digest }

```

Digest ::= OCTET STRING

- ⑦ 密文消息内容用 ASN.1 描述如下：

```

EncryptedData ::= SEQUENCE {
    version Version,
    encryptedContentInfo EncryptedContentInfo }

```

6. PKCS #8: Private-Key Information Syntax Standard (私钥信息语法标准)

PKCS #8 v1.2 描述私钥信息的语法格式。私钥信息包括私钥和一组属性。该标准还描述了私钥密文的语法，且允许使用基于口令的加密算法来加密私钥。在 RFC 5208 中又重新定义。

- ① 私钥信息格式用 ASN.1 描述如下：

```

PrivateKeyInfo ::= SEQUENCE {
    version Version,
    privateKeyAlgorithm PrivateKeyAlgorithmIdentifier,
    privateKey PrivateKey,
    attributes [0] IMPLICIT Attributes OPTIONAL }

```

Version ::= INTEGER

PrivateKeyAlgorithmIdentifier ::= AlgorithmIdentifier

PrivateKey ::= OCTET STRING

Attributes ::= SET OF Attribute

- ② 私钥密文格式用 ASN.1 描述如下：

```

EncryptedPrivateKeyInfo ::= SEQUENCE {
    encryptionAlgorithm EncryptionAlgorithmIdentifier,
    encryptedData EncryptedData }

```

EncryptionAlgorithmIdentifier ::= AlgorithmIdentifier

EncryptedData ::= OCTET STRING

7. PKCS #9: Selected Attribute Types (可供选择的属性类型)

PKCS #9 v2.0 定义了两个新的辅助对象类和精选的属性类型（基于这两个对象类）。

- ① 两个辅助对象类 pkcsEntity 和 naturalPerson 定义如下：

```

pkcsEntity OBJECT-CLASS ::= {
    SUBCLASS OF { top }
    KIND auxiliary
    MAY CONTAIN { PKCSEntityAttributeSet }
    ID pkcs-9-oc-pkcsEntity
}

```

PKCSEntityAttributeSet ATTRIBUTE ::= {

```

    pKCS7PDU | userPKCS12 | pKCS15Token | encryptedPrivateKeyInfo,
    .... For future extensions
}
naturalPerson OBJECT-CLASS ::= {
    SUBCLASS OF { top }
    KIND auxiliary
    MAY CONTAIN { NaturalPersonAttributeSet }
    ID pkcs-9-oc-naturalPerson
}
NaturalPersonAttributeSet ATTRIBUTE ::= {
    emailAddress | unstructuredName | unstructuredAddress |
    dateOfBirth | placeOfBirth | gender | countryOfCitizenship |
    countryOfResidence | pseudonym | serialNumber,
    .... For future extensions
}

```

② 基于 pkcsEntity 对象类的属性类型定义如下：

```

pKCS7PDU ATTRIBUTE ::= {
    WITH SYNTAX ContentInfo
    ID pkcs-9-at-pkcs7PDU
}
userPKCS12 ATTRIBUTE ::= {
    WITH SYNTAX PFX
    ID pkcs-9-at-userPKCS12
}
pKCS15Token ATTRIBUTE ::= {
    WITH SYNTAX PKCS15Token
    ID pkcs-9-at-pkcs15Token
}
encryptedPrivateKeyInfo ATTRIBUTE ::= {
    WITH SYNTAX EncryptedPrivateKeyInfo
    ID pkcs-9-at-encryptedPrivateKeyInfo
}

```

③ 基于 naturalPerson 对象类的属性类型定义如下：

```

emailAddress ATTRIBUTE ::= {
    WITH SYNTAX IA5String (SIZE (1..pkcs-9-ub-emailAddress))
    EQUALITY MATCHING RULE pkcs9CaseIgnoreMatch
    ID pkcs-9-at-emailAddress
}
unstructuredName ATTRIBUTE ::= {
    WITH SYNTAX PKCS9String {pkcs-9-ub-unstructuredName}
    EQUALITY MATCHING RULE pkcs9CaseIgnoreMatch
}

```

```

        ID pkcs-9-at-unstructuredName
    }
    unstructuredAddress ATTRIBUTE ::= {
        WITH SYNTAX DirectoryString {pkcs-9-ub-unstructuredAddress}
        EQUALITY MATCHING RULE caseIgnoreMatch
        ID pkcs-9-at-unstructuredAddress
    }
    dateOfBirth ATTRIBUTE ::= {
        WITH SYNTAX GeneralizedTime
        EQUALITY MATCHING RULE generalizedTimeMatch
        SINGLE VALUE TRUE
        ID pkcs-9-at-dateOfBirth
    }
    placeOfBirth ATTRIBUTE ::= {
        WITH SYNTAX DirectoryString {pkcs-9-ub-placeOfBirth}
        EQUALITY MATCHING RULE caseExactMatch
        SINGLE VALUE TRUE
        ID pkcs-9-at-placeOfBirth
    }
    gender ATTRIBUTE ::= {
        WITH SYNTAX PrintableString (SIZE (1) ^ FROM ("M" | "F" | "m" | "f"))
        EQUALITY MATCHING RULE caseIgnoreMatch
        SINGLE VALUE TRUE
        ID pkcs-9-at-gender
    }
    countryOfCitizenship ATTRIBUTE ::= {
        WITH SYNTAX PrintableString (SIZE (2) ^ CONSTRAINED BY {
-- Must be a two-letter country acronym in accordance with ISO/IEC 3166 --})
        EQUALITY MATCHING RULE caseIgnoreMatch
        ID pkcs-9-at-countryOfCitizenship
    }
    countryOfResidence ATTRIBUTE ::= {
        WITH SYNTAX PrintableString (SIZE (2) ^ CONSTRAINED BY {
-- Must be a two-letter country acronym in accordance with ISO/IEC 3166 --})
        EQUALITY MATCHING RULE caseIgnoreMatch
        ID pkcs-9-at-countryOfResidence
    }
    pseudonym ATTRIBUTE ::= {
        WITH SYNTAX DirectoryString {pkcs-9-ub-pseudonym}
        EQUALITY MATCHING RULE caseExactMatch
        ID id-at-pseudonym
    }

```

④ 用于 PKCS #7 的属性类型定义如下:

```

contentType ATTRIBUTE ::= {
    WITH SYNTAX ContentType
    EQUALITY MATCHING RULE objectIdentifierMatch
    SINGLE VALUE TRUE
    ID pkcs-9-at-contentType
}
ContentType ::= OBJECT IDENTIFIER
messageDigest ATTRIBUTE ::= {
    WITH SYNTAX MessageDigest
    EQUALITY MATCHING RULE octetStringMatch
    SINGLE VALUE TRUE
    ID pkcs-9-at-messageDigest
}
MessageDigest ::= OCTET STRING
signingTime ATTRIBUTE ::= {
    WITH SYNTAX SigningTime
    EQUALITY MATCHING RULE signingTimeMatch
    SINGLE VALUE TRUE
    ID pkcs-9-at-signingTime
}
SigningTime ::= Time -- imported from ISO/IEC 9594-8
randomNonce ATTRIBUTE ::= {
    WITH SYNTAX RandomNonce
    EQUALITY MATCHING RULE octetStringMatch
    SINGLE VALUE TRUE
    ID pkcs-9-at-randomNonce
}
RandomNonce ::= OCTET STRING (SIZE (4..MAX)) -- At least four bytes long
sequenceNumber ATTRIBUTE ::= {
    WITH SYNTAX SequenceNumber
    EQUALITY MATCHING RULE integerMatch
    SINGLE VALUE TRUE
    ID pkcs-9-at-sequenceNumber
}
SequenceNumber ::= INTEGER (1..MAX)
counterSignature ATTRIBUTE ::= {
    WITH SYNTAX SignerInfo
    ID pkcs-9-at-counterSignature
}

```

⑤ 用于 PKCS #10 的属性类型定义如下:

```
challengePassword ATTRIBUTE ::= {
    WITH SYNTAX DirectoryString {pkcs-9-ub-challengePassword}
    EQUALITY MATCHING RULE caseExactMatch
    SINGLE VALUE TRUE
    ID pkcs-9-at-challengePassword
}
extensionRequest ATTRIBUTE ::= {
    WITH SYNTAX ExtensionRequest
    SINGLE VALUE TRUE
    ID pkcs-9-at-extensionRequest
}
ExtensionRequest ::= Extensions
extendedCertificateAttributes ATTRIBUTE ::= {
    WITH SYNTAX SET OF Attribute
    SINGLE VALUE TRUE
    ID pkcs-9-at-extendedCertificateAttributes
}
```

- ⑥ 用于 PKCS #12 或 PKCS #15 的属性定义如下：

```
friendlyName ATTRIBUTE ::= {
    WITH SYNTAX BMPString (SIZE (1..pkcs-9-ub-friendlyName))
    EQUALITY MATCHING RULE caseIgnoreMatch
    SINGLE VALUE TRUE
    ID pkcs-9-at-friendlyName
}
localKeyId ATTRIBUTE ::= {
    WITH SYNTAX OCTET STRING
    EQUALITY MATCHING RULE octetStringMatch
    SINGLE VALUE TRUE
    ID pkcs-9-at-localKeyId
}
```

- ⑦ S/MIME 规范中定义的属性如下：

```
signingDescription ATTRIBUTE ::= {
    WITH SYNTAX DirectoryString {pkcs-9-ub-signingDescription}
    EQUALITY MATCHING RULE caseIgnoreMatch
    SINGLE VALUE TRUE
    ID pkcs-9-at-signingDescription
}
smimeCapabilities ATTRIBUTE ::= {
    WITH SYNTAX SMIMECapabilities
    SINGLE VALUE
    ID pkcs-9-at-smimeCapabilities
}
```

```

}
SMIMECapabilities ::= SEQUENCE OF SMIMECapability
SMIMECapability ::= SEQUENCE {
    algorithm ALGORITHM.&id ( {SMIMEv3Algorithms} ),
    parameters ALGORITHM.&Type ( {SMIMEv3Algorithms} {@algorithm} )
}
SMIMEv3Algorithms ALGORITHM ::= { ... -- See RFC 2633 -- }

```

8. PKCS #10: Certification Request Syntax Standard (证书请求语法标准)

PKCS #10 v1.7 描述了证书请求的语法格式。证书请求包括 DN 名称、公钥和一组可选属性，以及请求方对上述信息的签名。一个证书请求包括可辨别名、公开密钥和（可选的）一组属性，所有这些均由请求证书的用户签名。证书请求被发送给 CA，由 CA 基于证书请求中的内容签发数字证书。在 RFC 2986 中重新定义。

① 证书请求信息格式用 ASN.1 描述如下：

```

CertificationRequestInfo ::= SEQUENCE {
    version                INTEGER { v1 (0) } (v1,...) ,
    subject                Name,
    subjectPKInfo          SubjectPublicKeyInfo{ { PKInfoAlgorithms } },
    attributes              [0] Attributes{ { CRIAttributes } }
}
SubjectPublicKeyInfo { ALGORITHM : IOSet } ::= SEQUENCE {
    algorithm              AlgorithmIdentifier { { IOSet } },
    subjectPublicKey       BIT STRING
}
PKInfoAlgorithms ALGORITHM ::= {
    ... -- add any locally defined algorithms here -- }
Attributes { ATTRIBUTE:IOSet } ::= SET OF Attribute{ { IOSet } }
CRIAttributes ATTRIBUTE ::= {
    ... -- add any locally defined attributes here -- }
Attribute { ATTRIBUTE:IOSet } ::= SEQUENCE {
    type ATTRIBUTE.&id ( { IOSet } ),
    values SET SIZE (1..MAX) OF ATTRIBUTE.&Type ( { IOSet } {@type} )
}

```

② 证书请求格式用 ASN.1 描述如下：

```

CertificationRequest ::= SEQUENCE {
    certificationRequestInfo CertificationRequestInfo,
    signatureAlgorithm      AlgorithmIdentifier{ { SignatureAlgorithms } },
    signature                BIT STRING
}
AlgorithmIdentifier { ALGORITHM:IOSet } ::= SEQUENCE {
    algorithm ALGORITHM.&id ( { IOSet } ),

```

```

parameters ALGORITHM.&Type ( {IOSet} { @algorithm} ) OPTIONAL
}
SignatureAlgorithms ALGORITHM ::= {
... -- add any locally defined algorithms here -- }

```

9. PKCS #11: Cryptographic Token Interface Standard (密码 Token 接口标准)

PKCS #11 v2.2 定义了一种与密码设备 (如智能卡) 无关的编程接口技术。主要包括以下内容:

① 通用数据类型 (General Data Types)。主要分为 7 类。

A. 通用信息类型: CK_VERSION、CK_INFO、CK_NOTIFICATION;

B. Slot 及 Token 类型: CK_SLOT_ID、CK_SLOT_INFO、CK_TOKEN_INFO;

C. Session 类型: CK_SESSION_HANDLE、CK_USER_TYPE、CK_STATE、CK_SESSION_INFO;

D. 对象类型: CK_OBJECT_HANDLE、CK_OBJECT_CLASS、CK_HW_FEATURE_TYPE、CK_KEY_TYPE、CK_CERTIFICATE_TYPE、CK_ATTRIBUTE_TYPE、CK_ATTRIBUTE、CK_DATE;

E. 机制相关类型: CK_MECHANISM_TYPE、CK_MECHANISM、CK_MECHANISM_INFO;

F. 函数类型: CK_RV、CK_NOTIFY、CK_C_XXX、CK_FUNCTION_LIST;

G. Locking 相关类型: CK_CREATEMUTEX、CK_DESTROYMUTEX、CK_LOCKMUTEX、CK_UNLOCKMUTEX、CK_C_INITIALIZE_ARGS。

② 对象 (Objects)。

主要分为以下几类: 硬件特征对象、数据对象、证书对象、密钥对象、域参数对象和机制对象等。证书对象又分为 X.509 公钥证书对象、WTLS 公钥证书对象和 X.509 属性证书对象。密钥对象又分为公钥对象、私钥对象和秘密密钥对象。

③ 函数 (Functions)。

主要分为以下几类:

A. 通用功能函数。共 4 个: C_Initialize、C_Finalize、C_GetInfo、C_GetFunctionList;

B. Slot 及 Token 管理函数。共 9 个: C_GetSlotList、C_GetSlotInfo、C_GetTokenInfo、C_WaitForSlotEvent、C_GetMechanismList、C_GetMechanismInfo、C_InitToken、C_InitPIN、C_SetPIN;

C. Session 管理函数。共 8 个: C_OpenSession、C_CloseSession、C_CloseAllSessions、C_GetSessionInfo、C_GetOperationState、C_SetOperationState、C_Login、C_Logout;

D. 对象管理函数。共 9 个: C_CreateObject、C_CopyObject、C_DestroyObject、C_GetObjectSize、C_GetAttributeValue、C_SetAttributeValue、C_FindObjectsInit、C_FindObjects、C_FindObjectsFinal;

E. 加密函数。共 4 个: C_EncryptInit、C_Encrypt、C_EncryptUpdate、C_EncryptFinal;

F. 解密函数。共 4 个: C_DecryptInit、C_Decrypt、C_DecryptUpdate、C_DecryptFinal;

G. 消息摘要函数。共 5 个: C_DigestInit、C_Digest、C_DigestUpdate、C_DigestKey、C_DigestFinal;

H. 签名及 MAC 函数。共 6 个：C_SignInit、C_Sign、C_SignUpdate、C_SignFinal、C_SignRecoverInit、C_SignRecover;

I. 验证签名及 MAC 函数。共 6 个：C_VerifyInit、C_Verify、C_VerifyUpdate、C_VerifyFinal、C_VerifyRecoverInit、C_VerifyRecover;

J. 双功能 (dual-purpose) 密码函数。共 4 个：C_DigestEncryptUpdate、C_DecryptDigestUpdate、C_SignEncryptUpdate、C_DecryptVerifyUpdate;

K. 密钥管理函数。共 5 个：C_GenerateKey、C_GenerateKeyPair、C_WrapKey、C_UnwrapKey、C_DeriveKey;

L. 随机数生成函数。共 2 个：C_SeedRandom、C_GenerateRandom;

M. 并行函数管理函数。共 2 个：C_GetFunctionStatus、C_CancelFunction。

④ 机制 (Mechanisms)。

机制规定了特定的密码操作过程如何准确地执行。主要包括以下几类：

RSA、DSA、Elliptic Curve、Diffie-Hellman、KEA、Wrapping/unwrapping private keys、Generic secret key、HMAC mechanisms、RC2、RC4、RC5、AES、General block cipher、Key derivation by data encryption – DES & AES、Double and Triple-length DES、SKIPJACK、BATON、JUNIPER、MD2、MD5、SHA-1、SHA-256、SHA-384、SHA-512、FASTHASH、PKCS #5 and PKCS #5-style password-based encryption (PBE)、PKCS #12 password-based、encryption/authentication mechanisms、RIPE-MD、SET、LYNKS、SSL、TLS、WTLS、Miscellaneous simple key derivation mechanisms、CMS、Blowfish、Twofish。

10. PKCS #12: Personal Information Exchange Syntax Standard (个人信息交换语法标准)

PKCS #12 v1.0 描述了用于存储和传递个人身份信息的语法格式。个人身份信息包括私钥、证书、各种秘密及扩展等。支持本标准的机器设备、应用系统、浏览器、上网亭 (Internet Kiosk) 等应允许用户导入、导出和操作这种格式的个人身份信息。它的目标是为各种应用提供一个标准的单一密钥文件。常见的 PFX 文件就是遵循 PKCS#12 格式的文件。

个人身份信息格式用 ASN.1 描述如下：

```

PFX ::= SEQUENCE {
    version      INTEGER {v3 (3)} (v3,...) ,
    authSafe     ContentInfo,
    macData      MacData OPTIONAL
}
MacData ::= SEQUENCE {
    mac          DigestInfo,
    macSalt      OCTET STRING,
    iterations   INTEGER DEFAULT 1
    -- Note: The default is for historical reasons and its use is deprecated. A higher
    -- value, like 1024, is recommended.
}

```


11. PKCS #15: Cryptographic Token Information Format Standard (密码 Token 信息格式标准)

PKCS #15 v1.1 描述了存储于密码 Token 中密码凭证的一种格式标准, 允许密码令牌的用户向应用程序标识自己。此标准独立于 PKCS #11 接口或其他 API。RSA 已经放弃了这个标准的 IC 卡的相关部分, 并提交给了 ISO/IEC 7816-15。

29.2 ISO 7816 系列

ISO/IEC 7816 是电子识别卡或智能卡的国际标准, 由 ISO(International Organization for Standardization) 和 IEC(International Electrotechnical Commission) 组织共同维护, 目前包括 14 部分。

1. ISO 7816-1: Physical characteristics (卡的物理特性)

本部分规定了带触点集成电路卡的基本技术要求, 主要包括以下内容:

- ① 物理特性、记录方法、物理接口要求, 主要定义了该类卡的基本物理特性;
- ② 电气信号和传输协议, 规定了该类卡和终端间的电源、电气信号协议和信息交换协议, 涉及卡的信号频率、电压值、电流值、校验、操作规程和传输与通信协议。

2. ISO 7816-2: Cards with contacts: Dimensions and location of the contacts (触点集成电路卡: 触点的尺寸与位置)

本部分定义了 ID-1 类型集成电路卡每个触点的尺寸与位置, 同时提供了哪个标准定义了这些触点的信息。

3. ISO 7816-3: Cards with contacts: Electrical interface and transmission protocols (触点集成电路卡: 电信号和传输协议)

本部分规定了电源和信号的结构, 以及集成电路卡和接口设备(如终端)之间的信息交换。它还覆盖了信号速率、电压等级、电流值、奇偶约定、操作程序、传输机制及与卡的通信。但它不包括信息和指令的内容, 如发行人和使用者的识别、服务和限制、安全功能、日志记录和指令定义等。

4. ISO 7816-4: Organization, security and commands for interchange (用于交换的结构、安全和命令)

本部分规定了:

- ① 由接口设备至卡以及相反方向所发送的报文、命令和响应的内容;
- ② 获取卡内数据对象和元素的方式;
- ③ 在复位应答期间卡所发送的历史字节的结构及内容;
- ④ 当处理交换用的命令时, 在接口处所看到的文件和数据的结构;
- ⑤ 访问卡内文件和数据的方法;
- ⑥ 定义访问卡内文件和数据的权限的安全体系结构;
- ⑦ 安全报文交换的方法;

⑧ 访问卡所处理算法的方法。本标准不描述这些算法。

本规范没有涵盖卡内和/或外界的内部实现。

5. ISO 7816-5: Registration of application providers (卡应用提供者注册)

本部分定义了如何使用应用标识符(application identifier)确认卡上存在应用和从卡上获取应用。应用标识符标示卡中应用的元素,可以通过一个全局注册中心保证应用标识符的唯一性。

本标准确定了各种机构和规程,以确保和优化相应注册的可靠性。

6. ISO 7816-6: Interindustry data elements for interchange (行业间数据元)

本部分规定了在行业间使用的数据元素(包括复合数据元素),它定义了每个数据元素的以下特征:标识符、名称、说明和参考、格式和编码。

每个数据元素的布局以接口设备和卡之间可见的方式描述。

本文档提供了数据元素的定义而不考虑对它的任何使用限制。

本规范没有涵盖卡内和/或外界的内部实现。

7. ISO 7816-7: Interindustry commands for Structured Card Query Language (SCQL) (用于结构化卡查询语言(SCQL)的行业间命令)

本部分定义了:

① SCQL 数据库概念, SCQL (Structured Card Query Language) 表示结构化卡查询语言, 基于 SQL (参见 ISO 9075);

② 相关的行业间增强命令。

8. ISO 7816-8: Commands for security operations (与安全相关的行业间命令)

本部分规定了用于密码操作的行业间命令, 密码机制的选择和使用条件可能影响卡的输出, 算法和协议的适宜性评估在本标准范围之外。本部分规定了:

① 卡中使用的安全协议;

② 安全报文交换扩展;

③ 卡的安全功能/服务上的安全机制的映射, 包括卡内安全机制的描述;

④ 安全支持的数据元;

⑤ 在卡上实现的算法的使用(算法本身并不详细描述);

⑥ 证书的使用;

⑦ 与安全相关的命令。

本规范没有涵盖卡内和/或外界的内部实现, 并且不强制卡支持本部分描述的所有命令或命令的所有选项。

9. ISO 7816-9: Commands for card management (用于卡管理的命令)

本部分规定了用于卡管理和文件管理的行业间命令。这些命令覆盖卡的整个生命周期, 因此, 有些命令在卡发行到持卡人手中之前就被使用, 有些命令在卡终止后仍被使用。

本规范没有涵盖卡内和(或)外界的内部实现。

本规范的附件展示了如何控制加载数据(加载数据包括代码、密钥、小程序等)到卡中, 如可以通过校验加载实体的权限、以安全消息方式传输数据等。

10. ISO 7816-10: Electronic signals and answer to reset for synchronous cards (同步卡的电信号和复位应答)

本部分规定了在集成电路卡和接口设备（如终端）之间同步传输使用的电源、信号结构及复位应答结构。

本部分同时涵盖了信号速率、操作条件、与集成电路卡的通信。

本部分定义了 type-1 和 type-2 两种同步卡。

11. ISO 7816-11: Personal verification through biometric methods (通过生物识别方法的个人验证)

本部分规定了集成电路卡在行业间命令中使用生物识别方法验证个人的安全机制，它还定义了把卡作为生物参考数据或设备的数据结构和数据访问方法。

本标准不定义使用生物识别方法对人员的识别过程。

12. ISO 7816-12: Cards with contacts: USB electrical interface and operating procedures (带触点集成电路卡: USB 电气接口及操作规程)

本部分规定了提供 USB 接口的 IC 卡的操作条件，图 29-1 展示了 USB 接口的触点分配和此分配与 ISO/IEC7816-3 中使用的触点分配的相互关系。

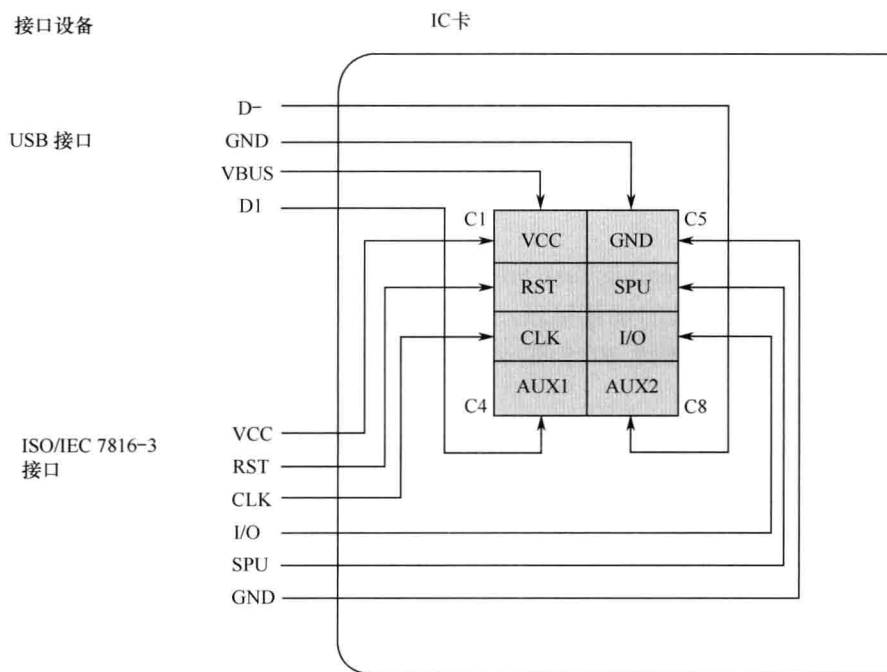


图 29-1 带有 USB 接口的 IC 卡的触点分配

13. ISO 7816-13: Commands for application management in multi-application environment (在多应用环境中用于应用管理的命令)

本部分规定了在多应用环境下用于应用管理的命令，这些命令涵盖了多应用环境下的应用的整个生命周期。可以在卡发行到持卡者手中之前或之后使用这些命令。

本规范没有涵盖卡内和/或外界的内部实现。

14. ISO 7816-15: Cryptographic information application (密码信息应用)

本部分规定了卡的一种应用, 该应用包含密码功能的信息。

本部分定义了用于密码信息的通用语法和格式, 以及在适当时共享该信息的机制。

本部分旨在:

- 便于运行于不同平台的各部分间的交互性 (平台无关);
- 使外部应用能利用多个制造商的产品和组件 (厂商无关);
- 使无需重写应用层的软件也能使用更先进的技术 (应用无关);
- 在维持现有的、相关的标准一致性的同时, 进行必要的和可行的扩展。

本部分支持下列功能:

- 在卡中存储密码信息的多个实例;
- 使用密码信息;
- 检索密码信息, 这一功能的关键因素在于“目录文件”的概念, 它提供了卡上对象和这些对象实际格式之间的一个间接层;
- 适当时候, 用 ISO/IEC 7816 其他部分中定义的数据对象来交叉引用密码信息;
- 不同的鉴别机制;
- 多个密码算法。

本规范没有涵盖卡内和/或外界的内部实现。

不强制要求执行本部分的所有选项。

29.3 IETF RFC 系列

在 IETF (Internet Engineering Task Force) 内有 PKIX (Public-Key Infrastructure (X.509)) 工作组, 负责与 X.509 有关的规范管理, PKIX 工作组从 1995 年 10 月 26 日开始启动, 到 2013 年 10 月 31 日关闭, 在近 20 年间, 发布了大量 RFC 规范。

1. RFC 2459

名称: Internet X.509 Public Key Infrastructure Certificate and CRL Profile。

发布日期: 1999-01。

状态: 被 RFC3280 替代。

简述: 介绍了应用于互联网 PKI 证书和证书撤销列表的格式和语义。描述了在互联网环境中证书路径的处理, 提供密码学算法的编码规则。

2. RFC 2510

名称: Internet X.509 Public Key Infrastructure Certificate Management Protocols。

发布日期: 1999-03。

状态: 被 RFC4210 替代。

简述: 描述了互联网 X.509 PKI 证书管理协议 (CMP), 定义了证书生成和管理所有相关方面的协议消息。

3. RFC 2511

名称: Internet X.509 Certificate Request Message Format。

发布日期: 1999-03。

状态: 被 RFC4211 替代。

简述: 描述了证书请求消息格式 (CRMF)。它被用来向 CA 传递一个产生 X.509 证书的请求 (可能通过 RA)。请求消息一般包括公钥和有关的注册信息。

4. RFC 2527

名称: Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework。

发布日期: 1999-03。

状态: 被 RFC3647 替代。

简述: 提供了一个帮助写作证书策略和证书业务规则的框架。特别地, 此框架提供了一系列需要在证书策略和证书业务规则中进行说明的综合主题。

5. RFC 2528

名称: Internet X.509 Public Key Infrastructure Representation of Key Exchange Algorithm (KEA) Keys in Internet X.509 Public Key Infrastructure Certificates。

发布日期: 1999-03。

简述: 描述了 X.509 v3 证书中包含 KEA (密钥交换算法) 密钥的格式和语义。说明了公钥信息和密钥用途扩展项的内容。

6. RFC 2559

名称: Internet X.509 Public Key Infrastructure Operational Protocols - LDAPv2。

发布日期: 1999-04。

状态: 被 RFC3494 替代。

简述: 描述了访问 PKI 信息库以获取信息和管理这些信息的需求, 它基于 LDAP 协议, 定义了使用和更新证书编码和证书撤销列表的协议框架。

7. RFC 2560

名称: X.509 Internet Public Key Infrastructure Online Certificate Status Protocol (OCSP)。

发布日期: 1999-06。

状态: 被 RFC6960 替代, 由 RFC6277 更新。

简述: 在线证书状态查询协议, 描述了无需证书撤销列表就可以决定一张数字证书当前状态的协议。

8. RFC 2585

名称: Internet X.509 Public Key Infrastructure Operational Protocols: FTP and HTTP。

发布日期: 1999-05。

简述: 描述了使用 FTP (File Transfer Protocol) 和 HTTP (Hypertext Transfer Protocol) 协议获取证书和证书撤销列表 (CRLs) 的方法。

9. RFC 2587

名称: Internet X.509 Public Key Infrastructure LDAPv2 Schema。

发布日期: 1999-06。

状态: 被 RFC4523 替代。

简述: 描述了以支持在 LDAPv2 环境中使用 PKI 而定义的最小 schema (语法)。

10. RFC 2797

名称: Certificate Management Messages over CMS。

发布日期: 2000-04。

状态: 被 RFC5272 替代。

简述: 定义了使用 CMS 进行证书管理的协议, 以满足基于 CMS 和 PKCS10 进行证书服务的接口需求, 以及满足基于 SMIMEV3 使用 DH (Diffie-Hellman) 公钥并用 DSA 签名的证书注册请求。

11. RFC 3029

名称: Internet X.509 Public Key Infrastructure Data Validation and Certification Server Protocols。

发布日期: 2001-02。

简述: 描述了通用数据验证和证书服务器及与之进行通信的协议, 数据验证和证书服务器可以认为是受信任的第三方机构, 用来构建不可否认服务。

12. RFC 3161

名称: Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)。

发布日期: 2001-08。

状态: 被 RFC5816 更新。

简述: 描述了发送到时间戳机构的请求数据格式和返回的响应数据格式, 并说明了安全需求。

13. RFC 3279

名称: Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile。

发布日期: 2002-05。

状态: 被 RFC4055、RFC4491、RFC5480、RFC5758 更新。

简述: 描述了应用于 X.509 公钥证书和证书撤销列表的算法和算法标识符。

14. RFC 3280

名称: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile。

发布日期: 2002-05。

状态: 被 RFC5280 替代, 被 RFC4325、RFC4630 更新。

简述: 替代 RFC2549。描述了 X.509 v3 证书和 X.509 v2 的证书撤销列表 (CRL) 在互

联网的使用。详细描述了 X.509 v3 证书格式、证书标准扩展，增加了与互联网相关的两项扩展，并指定了必要证书扩展。详细描述了 X.509 v2 的证书撤销列表格式和必要扩展。

15. RFC 3281

名称：An Internet Attribute Certificate Profile for Authorization。

发布日期：2002-05。

状态：被 RFC5755 替代。

简述：描述了基于授权的属性证书。

16. RFC 3379

名称：Delegated Path Validation and Delegated Path Discovery Protocol Requirements。

发布日期：2002-09。

简述：描述了公钥证书的委托路径验证（DPV）和委托路径发现（DPD）需求，以及 DPV 和 DPD 的管理策略需求。

17. RFC 3628

名称：Policy Requirements for Time-Stamping Authorities（TSAs）。

发布日期：2003-11。

简述：定义了基准时间戳策略需求，时间戳机构使用公钥证书发布时间戳标记，时间戳可精确到 1 秒或更小。

18. RFC 3647

名称：Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework。

发布日期：2003-11。

简述：替代 2527，增加了更多内容。提供了一个帮助写作证书策略和证书业务规则的框架。特别是此框架提供了一系列需要在证书策略和证书业务规则中进行说明的综合主题。

19. RFC 3709

名称：Internet X.509 Public Key Infrastructure: Logotypes in X.509 Certificates。

发布日期：2004-02。

状态：被 RFC6170 更新。

简述：描述了包含可视化信息（如图片、声音等）的证书扩展项。

20. RFC 3770

名称：Certificate Extensions and Attributes Supporting Authentication in Point-to-Point Protocol（PPP）and Wireless Local Area Networks（WLAN）。

发布日期：2004-05。

状态：被 RFC4334 替代。

简述：定义了两项 EAP（Extensible Authentication Protocol）扩展密钥用途，且包含 WLAN（Wireless LAN）系统服务标识的证书扩展项。

21. RFC 3779

名称: X.509 Extensions for IP Addresses and AS Identifiers。

发布日期: 2004-06。

简述: 定义了两个 X.509 v3 证书扩展项。第一项绑定 IP 地址列表或 IP 前缀, 第二项绑定自治系统标识列表。其目的是使证书使用者有权限访问这些系统和 IP 地址。

22. RFC 3820

名称: Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile。

发布日期: 2004-06。

简述: 描述了在互联网使用的代理证书格式。代理证书来源于实体证书, 相比实体证书, 代理证书在应用系统使用的权限受限。

23. RFC 4055

名称: Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile。

发布日期: 2005-06。

状态: 被 RFC5756 更新。

简述: 描述了 RSASSA-PSS (RSA Probabilistic Signature Scheme)、RSAES-OAEP (RSA Encryption Scheme - Optimal Asymmetric Encryption Padding) 算法, 以及应用于 PKCS#1 的哈希函数算法, 包括编码格式、算法标识符、参数格式。

24. RFC 4158

名称: Internet X.509 Public Key Infrastructure: Certification Path Building。

发布日期: 2005-09。

简述: 对指导开发者构建证书认证路径提供指南和建议。

25. RFC 4210

名称: Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)。

发布日期: 2005-09。

状态: 被 RFC6712 更新。

简述: 替代 RFC2510。描述了互联网 X.509 PKI 证书管理协议 CMP, 定义了 X.509 v3 证书生成和管理协议消息。CMP 提供了 PKI 组成部分之间的在线交互消息, 包括 CA 和客户系统之间的数据交换。

26. RFC 4211

名称: Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)。

发布日期: 2005-09。

简述: 替代 RFC2511。描述了证书请求消息格式 (CRMF)。它被用来向 CA 传递一个产生 X.509 证书请求 (可能通过 RA)。请求消息一般包括公钥和有关的注册信息。

27. RFC 4325

名称: Internet X.509 Public Key Infrastructure Authority Information Access Certificate

Revocation List (CRL) Extension。

发布日期：2005-12。

状态：被 RFC5280 替代。

简述：定义了 CRL 扩展项，即机构信息访问扩展项，此扩展项便于发现和获取 CRL 发布者证书。

28. RFC 4387

名称：Internet X.509 Public Key Infrastructure Operational Protocols: Certificate Store Access via HTTP。

发布日期：2006-02。

简述：描述了使用 HTTP/HTTPS 从 PKI 资源库中获取证书和证书撤销列表的接口。

29. RFC 4476

名称：Attribute Certificate (AC) Policies Extension。

发布日期：2006-05。

简述：定义了属性证书策略扩展项。

30. RFC 4630

名称：Update to DirectoryString Processing in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile。

发布日期：2006-08。

状态：被 RFC5280 替代。

简述：更新了对 DirectoryString 类型数据的处理方法。建议使用 UTF8String 和 PrintableString 类型数据。

31. RFC 4985

名称：Internet X.509 Public Key Infrastructure Subject Alternative Name for Expression of Service Name。

发布日期：2007-08。

简述：在 X.509 的使用者替代名扩展项中，对 otherName 字段定义了新名称格式，以允许证书使用者关联服务名称和 DNS 服务资源记录域名称。

32. RFC 5019

名称：The Lightweight Online Certificate Status Protocol (OCSP) Profile for High-Volume Environments。

发布日期：2007-09。

简述：定义了在线证书状态协议 (OCSP) 的服务框架，以解决在大规模（高容量）公钥基础设施使用 OCSP 的可扩展性问题，或在需要 PKI 的环境中提供轻量级解决方案，以尽量减少通信带宽和客户端处理。

33. RFC 5055

名称：Server-Based Certificate Validation Protocol (SCVP)。

发布日期：2007-12。

简述：描述了客户端委托服务端进行证书路径构建和证书路径验证，服务端依据验证策略进行证书路径构建和证书路径验证，以简化客户端实现和允许使用预定义的验证策略。

34. RFC 5272

名称：Certificate Management over CMS (CMC)。

发布日期：2008-06。

状态：被 RFC6402 更新。

简述：替代 RFC2797。定义了使用 CMS 进行证书管理的协议，以满足基于 CMS 和 PKCS10 进行证书服务的接口需求，以及满足仅加密密钥证书注册协议请求。

35. RFC 5280

名称：Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile。

发布日期：2008-05。

状态：被 RFC6818 更新。

简述：替代 RFC3280、RFC4325、RFC4630。描述了 X.509 v3 证书和 X.509 v2 的证书撤销列表 (CRL) 在互联网的使用。详细描述了 X.509 v3 证书格式及有关互联网名格式和语义的附加信息，并描述了证书标准扩展和两项与互联网相关的扩展。详细描述了 X.509 v2 证书撤销列表格式、标准扩展及互联网相关的扩展。

36. RFC 5480

名称：Elliptic Curve Cryptography Subject Public Key Information。

发布日期：2009-03。

简述：描述了使用者证书中支持椭圆曲线密码公钥信息的语法和语义。

37. RFC 5636

名称：Traceable Anonymous Certificate。

发布日期：2009-08。

简述：定义了一个实用的体系结构和协议，使证书使用者采用化名方式的 X.509 证书以提供身份保密性，又能保证需要时把此证书映射到真正的用户。

38. RFC 5697

名称：Other Certificates Extension。

发布日期：2009-11。

简述：定义了一个扩展项，使一个实体的一系列证书能够关联起来，在检索应用信息时，可以使用此数据而不需要考虑证书的更新。

39. RFC 5755

名称：An Internet Attribute Certificate Profile for Authorization。

发布日期：2010-01。

简述：替代 RFC3281。描述了基于授权的属性证书。

40. RFC 5758

名称：Internet X.509 Public Key Infrastructure: Additional Algorithms and Identifiers for DSA and ECDSA。

发布日期：2010-01。

简述：更新 RFC3279。描述了当使用 SHA-224、SHA-256、SHA-384 或 SHA-512 作为哈希算法时，DSA 和 ECDSA 签名算法的标识符和 ASN.1 编码规则。适用于签发 X.509 证书和证书撤销列表签名。

41. RFC 5816

名称：ESSCertIDv2 Update for RFC 3161。

发布日期：2010-04。

简述：更新了 RFC3161。允许使用在 RFC5035 定义的 ESSCertIDv2 数据类型引用签名者证书哈希值。

42. RFC 5912

名称：New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX)。

发布日期：2010-06。

状态：被 RFC6960 更新。

简述：更新 PKIX 证书格式及关联格式的 ASN.1 编码与 2002 年版本的 ASN.1 标准一致。没有更改生成后的实际数据格式，只是更改了语法。

43. RFC 6025

名称：ASN.1 Translation。

发布日期：2010-10。

简述：提供从一个 ASN.1 版本转换到另一个 ASN.1 版本模块的语法指南，以规范 ASN.1 模块作者和实现者。

44. RFC 6170

名称：Internet X.509 Public Key Infrastructure—Certificate Image。

发布日期：2011-05。

简述：更新 RFC3709。通过定义一个新的 RFC3709 otherLogos 类型，以可视化方式绑定证书图像到公钥证书中。

45. RFC 6277

名称：Online Certificate Status Protocol Algorithm Agility。

发布日期：2011-06。

状态：被 RFC6960 替代。

简述：规定了 OCSP 服务端签名算法选择规则及扩展，以允许客户端通知服务端其支持的特定签名算法。

46. RFC 6402

名称: Certificate Management over CMS (CMC) Updates。

发布日期: 2011-11。

简述: 更新了 RFC5272、RFC5273 和 RFC5274。包含一组对 CMC 基础语法的更新。
新项目有: 定义服务端密钥产生策略的新控制, 定义使用者信息访问值以标识 CMC 服务端, 注册运行 CMC 服务的 TCP/IP 端口号。

47. RFC 6712

名称: Internet X.509 Public Key Infrastructure—HTTP Transfer for the Certificate Management Protocol (CMP)。

发布日期: 2012-09。

简述: 更新了 RFC4210。描述了如何通过 HTTP 协议承载 CMP (证书管理协议)。

48. RFC 6818

名称: Updates to the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile。

发布日期: 2013-01。

简述: 更新 RFC5280。本文档更新了证书策略中 explicitText 字段可接受编码方法, 以及转换国际域名标签到 ASCII 的规则。提供了一些对使用自签名证书、信任链和一些安全注意事项的说明。

49. RFC 6960

名称: X.509 Internet Public Key Infrastructure Online Certificate Status Protocol (OCSP)。

发布日期: 2013-06。

简述: 替代 RFC2560 和 RFC6277, 更新 RFC5912。描述了在不需要 CRL (证书撤销列表) 的情况下确定当前证书状态的有用协议。

50. RFC 7030

名称: Enrollment over Secure Transport。

发布日期: 2013-10。

简述: 描述了使用 CMC (Certificate Management over CMS) 通过安全传输进行客户端证书注册的框架。

29.4 Microsoft 规范

在 PKI 应用中, 使用较多的是 CryptoAPI 或 CNG 接口库。

1. CryptoAPI

Windows CryptoAPI 是微软公司提出的安全加密应用服务框架, 它提供了在 Win32 环境下使用认证、编码、加密和签名等安全服务的标准加密接口, 用于增强应用程序的安全性与可控性。应用开发者在不了解复杂的加密机制和加密算法的情况下, 可以简便、快速

地开发出标准、通用和易于扩展的安全加密应用程序。

CryptoAPI 是一组函数，为了完成数学计算，必须具有密码服务提供者模块 (Cryptographic Service Provider, CSP)。Microsoft 通过 RSA Base Provider 在操作系统级提供一个 CSP，使用 RSA 公钥加密算法，更多的 CSP 可以根据需要增加到应用中。事实上，CSP 有可能与硬件设备（如智能卡）一起来进行数据加密。CryptoAPI 接口允许通过简单的函数调用来加密数据、交换公钥、哈希一个消息来建立摘要以及生成数字签名。CryptoAPI 还为许多高级安全性服务提供了密码操作，包括用于加密客户机/服务器消息，用于在各个平台之间传递机密数据和密钥的 PFX、代码签名等。

CryptoAPI 体系共由 5 部分组成，体系结构如图 29-2 所示。

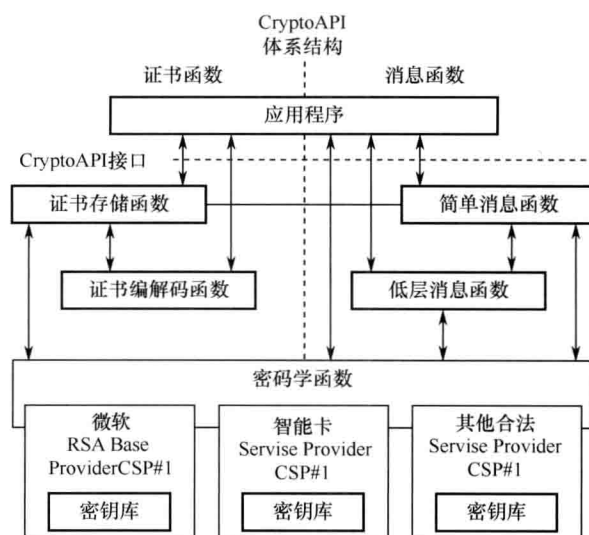


图 29-2 CryptoAPI 体系结构

(1) 基本加密函数 (Base Cryptographic Functions)

用于选择 CSP、建立 CSP 连接、产生密钥、交换及传输密钥等操作。

(2) 证书编解码函数 (Certificate Encode/Decode Functions)

用于数据加密、解密、哈希等操作。这类函数支持数据的加密/解密操作；计算哈希、签发和验证数字签名操作；实现证书、证书撤销列表、证书请求和证书扩展等编码和解码操作。

(3) 证书库管理函数 (Certificate Store Functions)

用于数字证书及证书库管理等操作。这组函数用于管理证书、证书撤销列表和证书信任列表的使用、存储、获取等。

(4) 简单消息函数 (Simplified Message Functions)

用于消息处理，比如消息编码/解码、消息加/解密、数字签名及签名验证等操作。它是把多个低层消息函数包装在一起以完成某个特定任务，方便用户使用。

(5) 低层消息函数 (Low-level Message Functions)

低层消息函数对传输的 PKCS#7 数据进行编码，对接收到的 PKCS#7 数据进行解码，并且对接收到的消息进行解码和验证。它可以实现简单消息函数的所有功能，且提供更大的灵活性，但需要更多的函数调用。

每类函数的命名前缀都有约定，前缀约定如下：基本加密函数 **Crypt**、证书编码与解码函数 **Crypt**、证书库管理函数 **Store**、简单消息函数 **Message**、低层消息函数 **Msg**。

2. CNG

Windows Vista 引入了新的加密 API 以替代旧的 **CryptoAPI**。旧的 **CryptoAPI** 存在于早期版本的 Windows NT 系列和 Windows 95 中。下一代加密技术（CNG）旨在长期替代 **CryptoAPI**，取代 **CryptoAPI** 提供的所有加密基元或服务。CNG 支持 **CryptoAPI** 提供的所有算法，而且应用更广泛并且包括许多新算法和更灵活的设计，从而为开发人员提供了对如何执行加密操作，以及算法如何协同工作以执行各种操作的更强的控制能力。

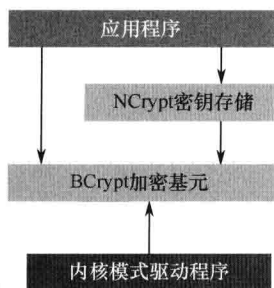


图 29-3 CNG 体系结构

图 29-3 说明了 CNG 的总体设计。**BCrypt** 是 CNG 的子集，提供加密基元，如随机数字生成、哈希函数、签名和加密密钥。**NCrypt** 也是 CNG 的子集，但它提供密钥存储工具，以支持永久保存非对称密钥和硬件（如智能卡）。从 **BCrypt** 和 **NCrypt** 的命名并不能了解太多内容。**BCrypt** 仅仅是头文件和为 CNG 提供基本服务的 DLL 的名称。在这种情况下，“B”代表“base”。同样，**NCrypt** 仅仅是头文件和提供高级别密钥存储功能的 DLL 的名称。“N”代表“new”。

29.5 Java 安全 API 规范

按照 Java 安全白皮书，Java 安全体系基本上分为 5 部分，分别是 Java 平台、Java 密码体系、Java 认证与授权、安全通信、PKI（公钥密码基础设施）。Java 安全技术包括一系列的 API、工具，以及常用密码算法、机制和协议。

1. Java 平台

Java 语言本身嵌入了安全特性，其中包括编译器和 JVM 对强数据类型的支持、自动内存管理、字节代码的验证机制以及独特的安全类加载方式，这些特性都是 Java 语言本身所具备的。

2. Java 密码体系

Java 密码体系依赖于 JCA 和 JCE，是两个非常重要的框架，它们提供了非常简洁通用的 API 接口。接口跟实现是完全分离的，即 Java 开发者可以采用 Sun 的实现方式，也可以接受其他的实现方式。

3. Java 认证与授权

Java 认证与授权用于两个目的：

- ① 针对用户认证，用于安全可靠地确定是谁当前正在执行 Java 代码。
- ② 针对用户授权，用于确保他们拥有所执行动作的访问控制权限。

JAAS（Java Authentication Authorization Service，Java 认证和授权 API）以插件方式工作，从而允许应用程序与底层身份验证技术相隔离。新的或更新的验证技术可以插入应用程序，而不需要修改应用程序本身。

4. 安全通信

主要规范了标准安全通信协议（SSL、TLS、Kerberos、SASL 等）的 API 和实现。应用最多的是 SSL/TLS，其次是 Kerberos。

5. PKI（Public Key Infrastructure，公钥密码基础设施）体系

Java PKI 规范提供了管理密钥和证书的 API，实现的协议包括：X.509 规范、CRL（证书作废列表）、PKCS#11、PKCS#12、PKIX（RFC3280）、在线证书状态协议（OCSP）。

API 接口规范可分为多类，具体如表 29-1 所示。

表 29-1 Java API 接口分类

分类	Java 包
通用安全相关包	java.security javax.crypto java.security.cert java.security.spec javax.crypto.spec java.security.interfaces javax.crypto.interfaces java.security.cert
JAAS 相关包	javax.security.auth javax.security.auth.callback javax.security.auth.kerberos javax.security.auth.login javax.security.auth.spi javax.security.auth.x500 com.sun.security.auth com.sun.security.auth.callback com.sun.security.auth.login com.sun.security.auth.module
GSS-API 相关包	org.ietf.jgss com.sun.security.jgss
JSSE 相关包	javax.net javax.net.ssl
SASL 相关包	javax.security.sasl
基于 SSL/TLS 的 RMI 套接字包	javax.rmi.ssl
XML 数字签名相关包	javax.xml.crypto javax.xml.crypto.dom javax.xml.crypto.dsig javax.xml.crypto.dsig.dom javax.xml.crypto.dsig.keyinfo javax.xml.crypto.dsig.spec
智能卡 I/O 包	javax.smartcardio

29.6 CCID 规范

CCID (Integrated Circuit (s) Cards Interface Devices) 是由几大国际级 IT 企业共同制定的一个标准, 它提供了一种智能卡读写设备与主机或其他嵌入式主机实现相互通信的通用机制。可以从 <http://www.usb.org> 网站下载规范版本。

CCID 标准规定了 CCID 设备是一种芯片/智能卡接口设备, 设备通过 USB 接口与主机或其他嵌入式主机连接, 进行符合 CCID 标准的数据通信。同时设备通过符合 ISO/IEC 7816 标准协议的接口与智能卡进行通信, 其结构如图 29-4 所示。

CCID 为所有通过 USB 连接的智能卡读写器定义了一个标准通信协议, 使得相同的主机端驱动可以与任何符合 CCID 标准的智能卡读写器进行通信。

微软公司在其 Windows 2000 及以下的操作系统中提供并支持 CCID 驱动, 使设备生产厂商可以轻松地开发使用符合 CCID 接口标准的设备。同时, CCID 接口标准支持 PC/SC 接口调用。在其他开源操作系统如 Linux 的众多版本上, 也有许多开源的 CCID 驱动可供开发者和使用者使用。

CCID 与 PC/SC 的关系可以理解为载体与被载体的关系。PC/SC 是一个大的框架, CCID 只是针对 USB 通信的一个协议。PC/SC 是一个让卡片懂得用户需要卡片进行操作的一个协议, 在通信中充当了 CCID 协议的消息内容。CCID 是让读卡器懂得 PC 发出的命令, 从而规范其命令的一个协议, 在通信中充当了 ISO/IEC-7816 命令的载体。

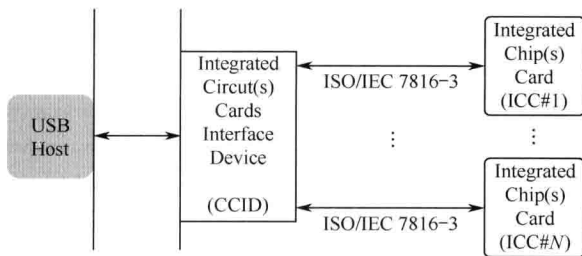


图 29-4 CCID 结构示意图

附录 主要参考资料

- [1] 维基百科网站, zh.wikipedia.org, en.wikipedia.org
- [2] OpenLdap 网站, www.openldap.org
- [3] IETF RFC 网站, www.ietf.org/rfc
- [4] CSDN 博客网站, blog.csdn.net
- [5] Oracle 技术网站, www.oracle.com/technetwork
- [6] Microsoft 开发者网站, msdn.microsoft.com
- [7] 163 博客网站, <http://www.blog.163.com/>
- [8] OpenSSL 网站, <http://www.openssl.org>
- [9] Apache 网站, <http://httpd.apache.org>
- [10] Tomcat 网站, <http://tomcat.apache.org>
- [11] EJBCA 网站, <http://www.ejbca.org>
- [12] ItEye 网站, www.iteye.com
- [13] Codeguru 网站, www.codeguru.com
- [14] 北京数字证书认证中心网站, www.bjca.org.cn
- [15] 中国金融认证中心网站, www.cfca.com.cn
- [16] 东方中讯数字证书认证网站, www.ezca.org
- [17] 工信部电子认证业务规则规范网站, xxaqs.miit.gov.cn
- [18] 国家商用密码管理办公室网站, www.oscca.gov.cn
- [19] 卫生部网站, wsb.moh.gov.cn
- [20] 互动百科网站, www.baike.com
- [21] IC 卡标准-ISO-IEC 7816 (1~15)
- [22] Smart-Card Integrated Circuit(s) Card Interface Devices, www.usb.org
- [23] 公钥密码标准, Public Key Cryptography Standard, PKCS#1~PKCS#15.
- [24] 智能 IC 卡及智能密码钥匙密码应用接口规范.
- [25] ZHANG Mingde etc. Research on Model of Trust Degrees for PKI.IAS 2009, ISBN 978-0-7695-3744-3, Aug. 2009, vol. 2: pp.647-650.
- [26] ZHANG Mingde etc. Improved Approach on Modeling and Reasoning about PKI/WPKI. WiCOM 2010, ISBN: 978-1-4244-3709-2, Sep. 2010: pp.1-4.
- [27] 张明德等. 改进的 PKI 可信度模型. 小型微型计算机系统, 2012 年 2 月, 第 33 卷第 2 期, P370-375.
- [28] 张明德等. 应用安全中身份认证建模及推理方法. 小型微型计算机系统, 2012 年 4 月, 第 33 卷第 4 期, P754-758.
- [29] 张明德等. 身份认证可信度研究. 计算机科学, 2011 年 11 月, 第 38 卷第 11 期, P43-47.

- [30] 张明德等. PKI 技术发展趋势浅析. 金融电子化, 2005 年 11 月, P65-P68.
- [31] 张明德等. 应用安全模型研究. 信息网络安全, 2012 年第 8 期, P121-125.
- [32] 张明德. 应用安全机制研究. 信息网络安全, 2013 年增刊, P17-20.
- [33] 周永彬等. 一种高效和可扩展的 OCSP 系统. 通信学报, 2003 (11): 94-99.
- [34] 上海信息安全工程技术研究中心, 国家信息安全工程技术研究中心. 简易在线证书状态协议 SOCSP 标准 (报批稿). 2009.
- [35] A Layman's Guide to a Subset of ASN.1, BER, and DER. Revised November 1, 1993.
- [36] 梁栋. Java 加密与解密的艺术. 北京: 机械工业出版社, 2010.
- [37] 张明德等. 商业银行密码技术应用. 北京: 电子工业出版社, 2011.
- [38] 王善平. 古今密码学趣谈. 北京: 电子工业出版社, 2012.
- [39] 宁宇鹏等. PKI 技术. 北京: 机械工业出版社, 2004.
- [40] 高海英等. VPN 技术. 北京: 机械工业出版社, 2004.
- [41] 马臣云等. 精通 PKI 网络安全认证技术与编程实现. 北京: 人民邮电出版社, 2008.
- [42] Bruce Schneier. 应用密码学. 吴世忠等译. 北京: 机械工业出版社, 2000.
- [43] Timothy A. Howes Ph.D. etc. Understanding and Deploying LDAP Directory Services, Second Edition.
- [44] 李婷. 智能化 DER 编码系统的设计与实现. 北京化工大学硕士学位论文, 2000. 5.
- [45] 戚翔. 基于 IPSec 的动态 VPN 技术的研究. 北京科技大学硕士学位论文, 2006. 12.
- [46] 黄华健. SSLVPN 中安全身份认证的研究. 西安电子科技大学硕士学位论文, 2008. 1.
- [47] 密码行业系列标准, GM/T 0001-0038.
- [48] JR/T 0068-2012 网上银行系统信息安全通用规范 (试行).
- [49] GB/T 25056-2010 信息安全技术 证书认证系统密码及其相关安全技术规范.
- [50] GB/T 28447-2012 信息安全技术 电子认证服务机构运营管理规范.
- [51] 深圳市前瞻商业资讯有限公司. 2011—2015 年中国电子认证服务业发展前景与投资预测分析报告.
- [52] 梁琼文等. 简易在线证书状态协议 SOCSP 的分析与改进.
- [53] FIPS 140-2 Security requirements for cryptographic modules.

反侵权盗版声明

电子工业出版社依法对本作品享有专有出版权。任何未经权利人书面许可，复制、销售或通过信息网络传播本作品的行为；歪曲、篡改、剽窃本作品的行为，均违反《中华人民共和国著作权法》，其行为人应承担相应的民事责任和行政责任，构成犯罪的，将被依法追究刑事责任。

为了维护市场秩序，保护权利人的合法权益，我社将依法查处和打击侵权盗版的单位和个人。欢迎社会各界人士积极举报侵权盗版行为，本社将奖励举报有功人员，并保证举报人的信息不被泄露。

举报电话：(010) 88254396；(010) 88258888

传 真：(010) 88254397

E-mail: dbqq@phei.com.cn

通信地址：北京市万寿路 173 信箱

电子工业出版社总编办公室

邮 编：100036